FH·W-S

iOS Programmierung

(mit Swift)

Peter Braun, Florian Bachmann & Andreas Wittmann
@pe_braun @florianbachmann @anwittmann

Deutsche Telekom AG
FHWS - Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt

#FHWSSwift

Agenda

1. **Introduction** – Organisatorisches
2. **First iOS-Project** – Hello World, **First iOS-Project** – Still Hello World (now with Code 😄)
3. **Swift,** Wait!, What about Objective-C?, Why Swift?
4. **A (not so) Quick Tour**
5. **Documentation**
6. **The basics –** iOS Architecture & more
7. **User Interfaces –** View Controller, Autolayout & Size Classes
8. **Storyboard & Segues**
9. **Tables & NavigationController**
10. **TabBarController**
11. **Notifications**
12. **Picker Views**
13. **Touches, Gestures, 3D Touch, Peek & Pop**
14. **StackViews**
15. **Networking** – JSON & Dependency Managers
16. **WebKit**
17. **Maps**
18. **Storage & Data persistency –** NSUserDefaults, NSKeyedArchiver & Core Data
19. **ObjC**

The Basics
iOS Architecture & more

Platform Components - Tools

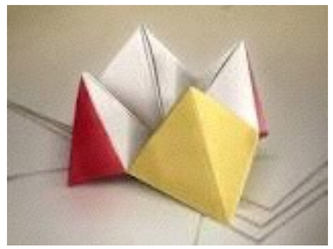
Xcode

Betas FTW! :)

iOS Simulator

Instruments

AppCode



Reveal

Spark

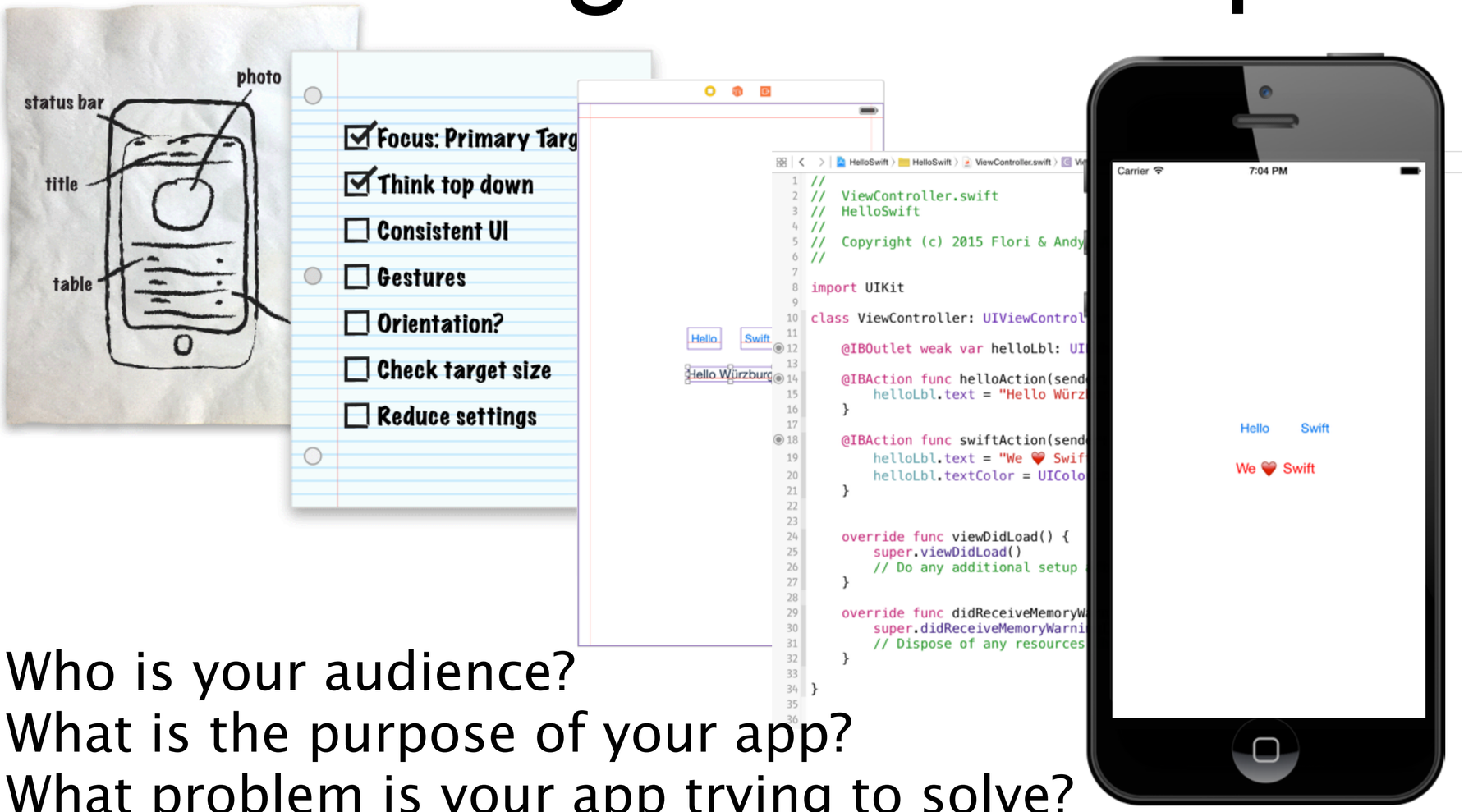
Dash

PaintCode

SimPholders

Alcatraz

Defining the concept



1. Who is your audience?
2. What is the purpose of your app?
3. What problem is your app trying to solve?
4. What content will your app incorporate?

https://developer.apple.com/videos/wwdc/2014/#223

Source: Prototyping: Fake It Till You Make It

Source: Apple - Start Developing iOS Apps Today

Golden Rules of Interface Design

The eight by Ben Shneiderman

1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialog to yield closure.
5. Prevent errors
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

https://www.google.de/search?q=golden+rules+of+interface+design

Golden Rules of Interface Design
Andy & Flori

1. Keep it simple! It is an App!
2. Our mothers need to understand it
3. Screens are small, fingers are thick
4. Focus on the key feature
5. one App per use case!
6. Gestures are still new, explain them
7. make it conform to the HIG

HIG - Human Interface Guidelines

On This Page

Designing for iOS

iOS embodies the following themes:

- **Deference.** The UI helps people understand and interact with the content, but never competes with it.
- **Clarity.** Text is legible at every size, icons are precise and lucid, adornments are subtle and appropriate, and a sharpened focus on functionality motivates the design.
- **Depth.** Visual layers and realistic motion impart vitality and heighten people's delight and understanding.

HIG

Mobile - Focus

- Memory is limited
- Only one App at a time
- It is a phone, not an iMac (small)
- keep restrictions in mind
- Only one app at a time
- Context is key (do one thing and do it well)
- Data is stored per app
- Data exchange between apps is hard
- minimized Text input (despite developing a Text editor)
- information flow from top to bottom

iOS Screen Sizes

iOS Res

A quick reference for iOS devices.

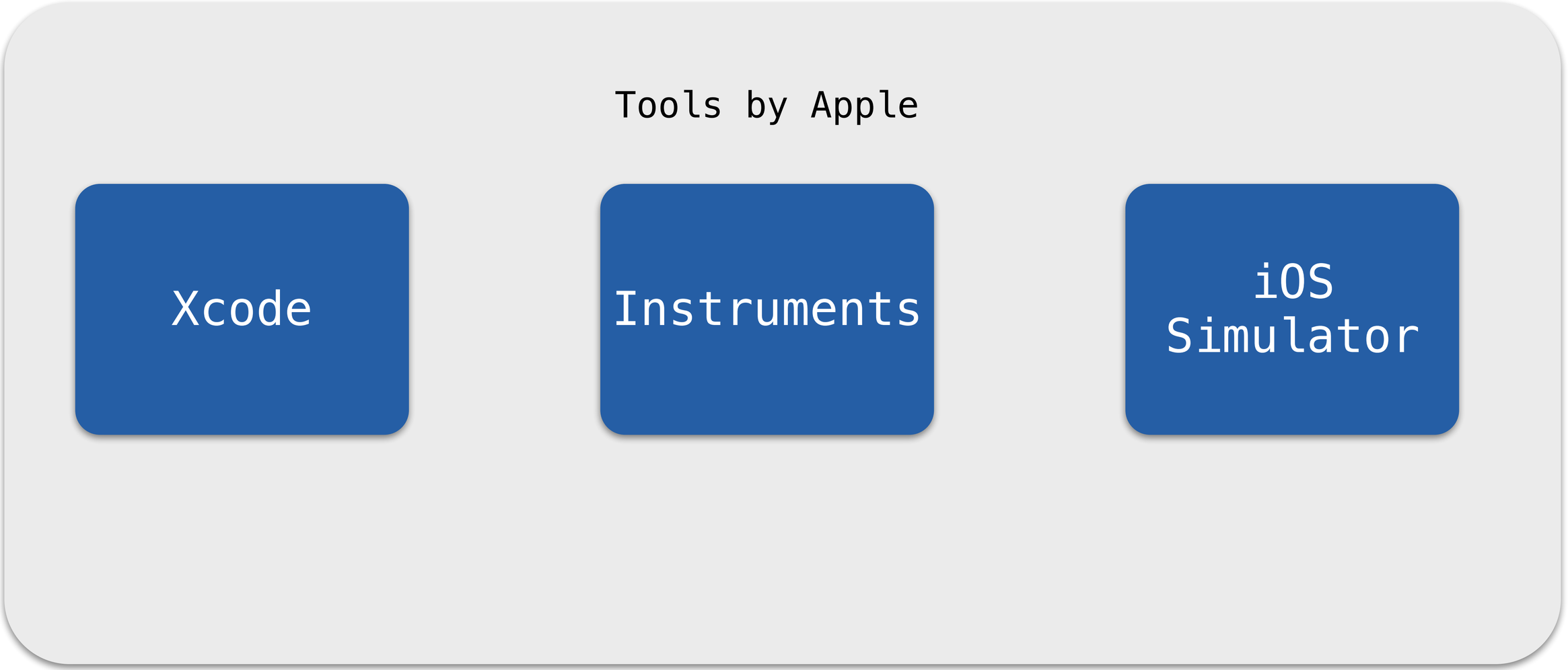
FILTER	iPhone 5	iPhone 6	iPhone 6 Plus	iPad Mini	iPad Air	iPad Pro	Watch 38mm	Watch 42mm
RESOLUTION	640 x 1136	750 x 1334	1242 x 2208 ⓘ	768 x 1024	1536 x 2048	2048 x 2732	272 x 340	312 x 390
LOGICAL RESOLUTION	320 x 568	375 x 667	414 x 736	768 x 1024	768 x 1024	1024 x 1366? ⓘ		
ASSOCIATED DEVICES	iPhone 5, 5C, 5S, iPod Touch 5g	iPhone 6, iPhone 6s	iPhone 6 Plus, iPhone 6s Plus	iPad Mini 2, iPad Mini 3, iPad Mini 4	iPad 3, iPad 4, iPad Air, iPad Air 2			
PPI	326 ppi	326 ppi	401 ppi	264 ppi	264 ppi	264 ppi	326 ppi	326 ppi
ASPECT RATIO	9:16	9:16	9:16	3:4	3:4	3:4	4:5	4:5

file@**3x**.png

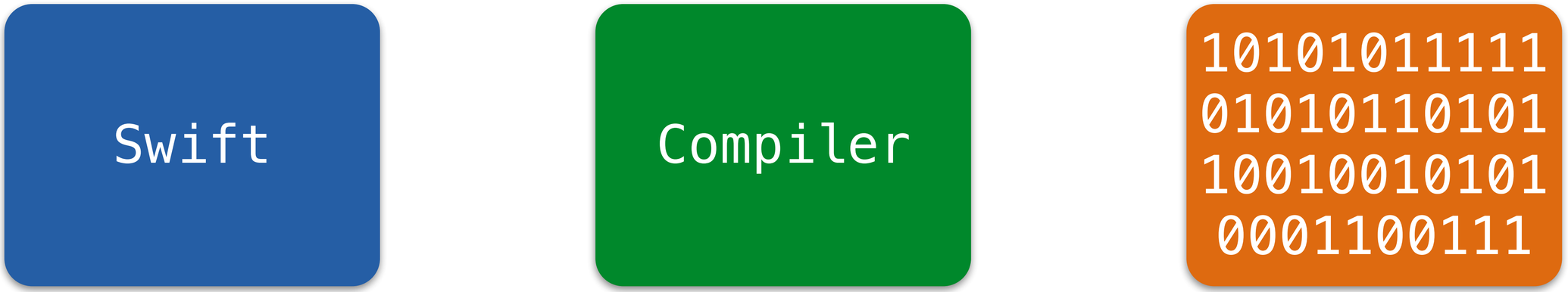
file@**2x**.png

Source: http://www.iosres.com/

App Development



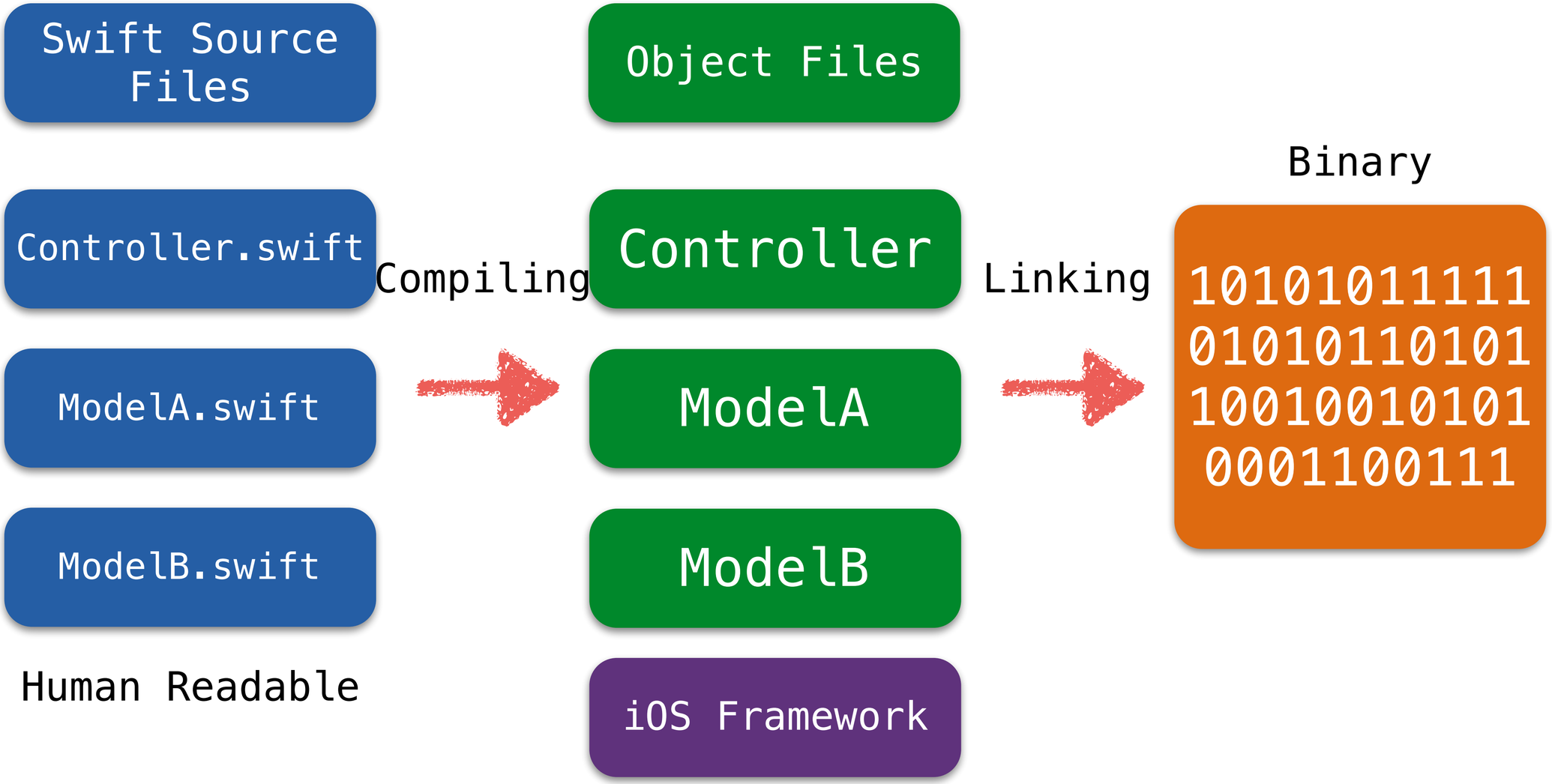
App Compling Steps



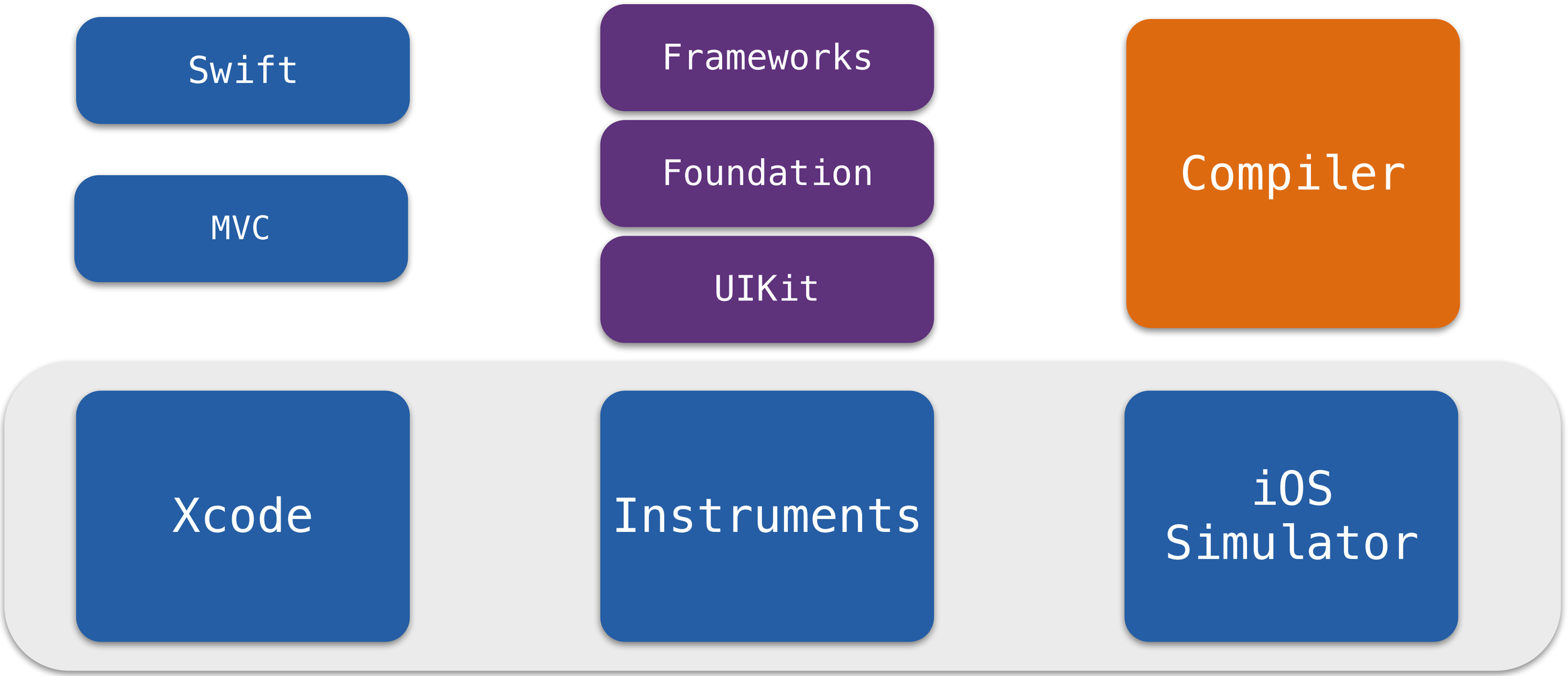
Human Readable

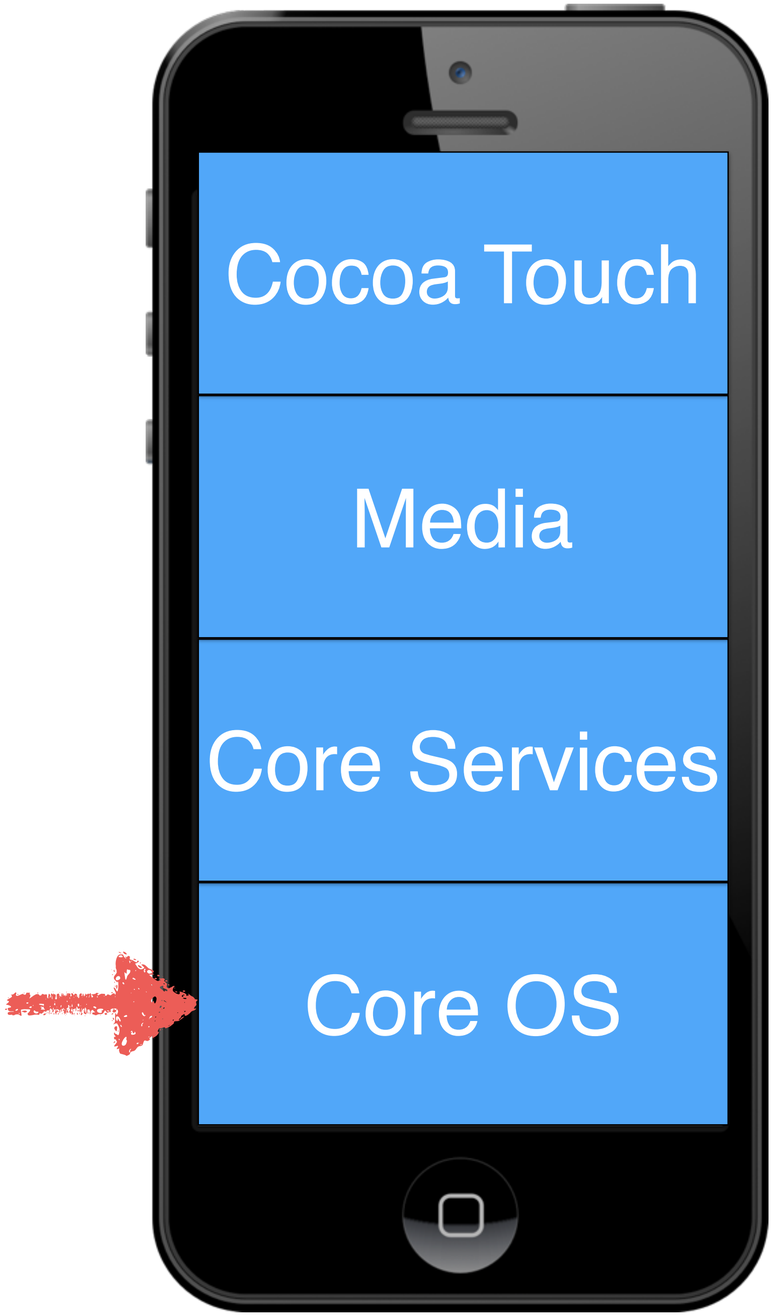
machine code

App Compling Steps



App Ingredients

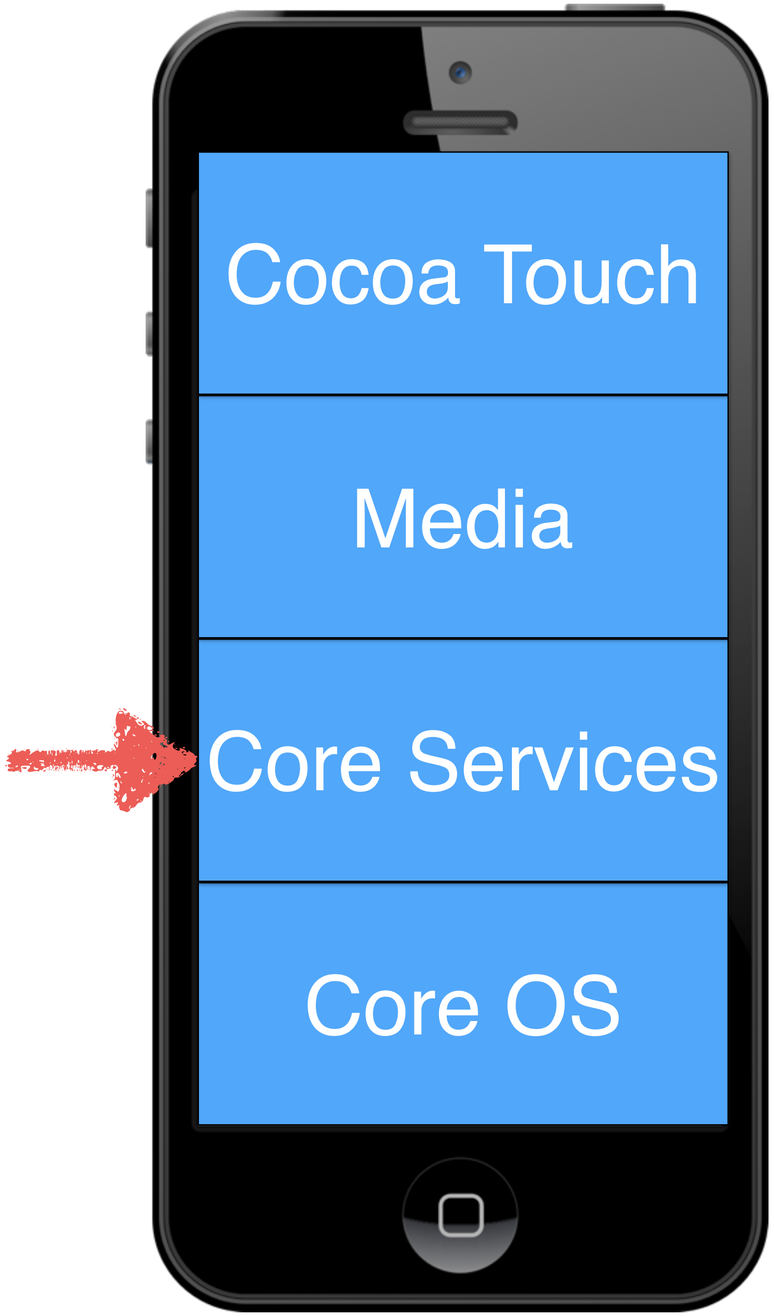




What is iOS
1. Core OS Layer

OSX Kernel	Power Management
Mach 3.0	Keychain Access
BSD	Certificates
Sockets	Files System
Security	Bonjous

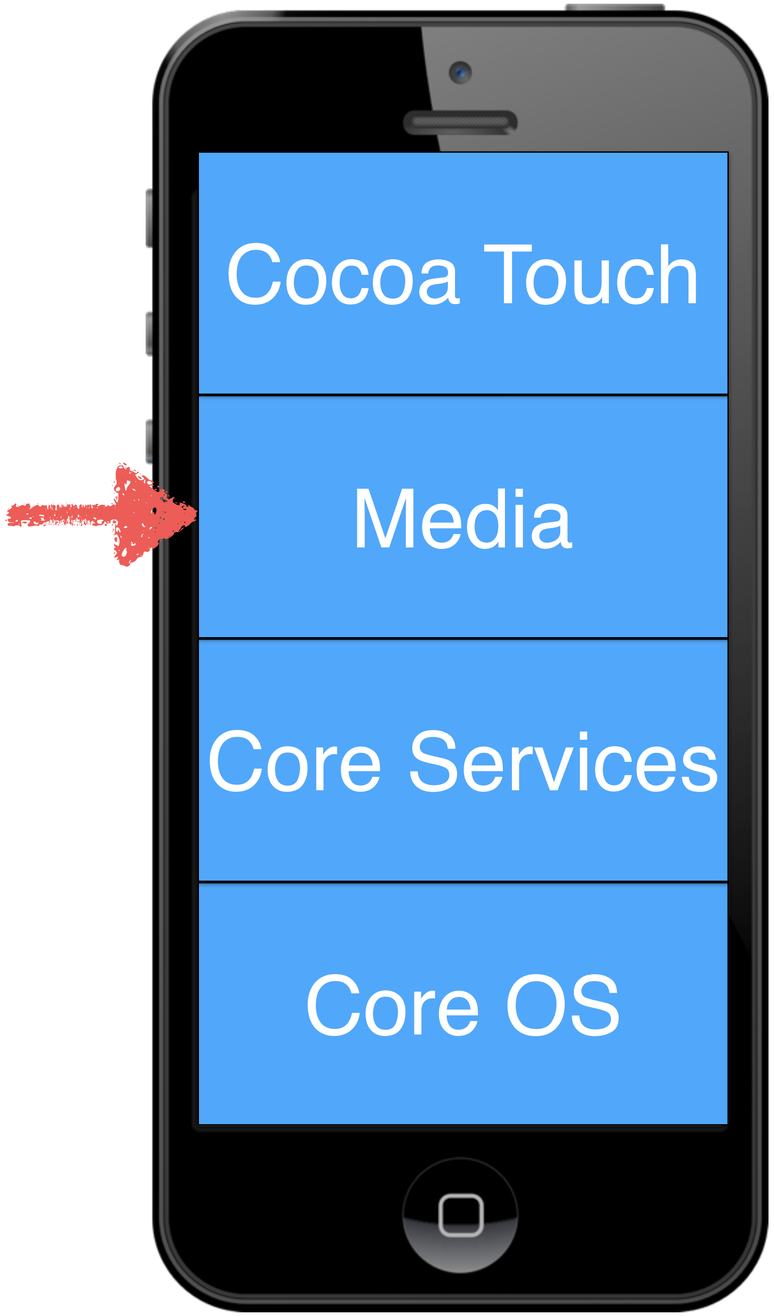
Source: Inspired by Stanford - CS193p



What is iOS
2. Core Services Layer

Collection	Core Location
Address Book	Net Services
Networking	Threading
File Access	Preferences
SQLite	URL Utilities

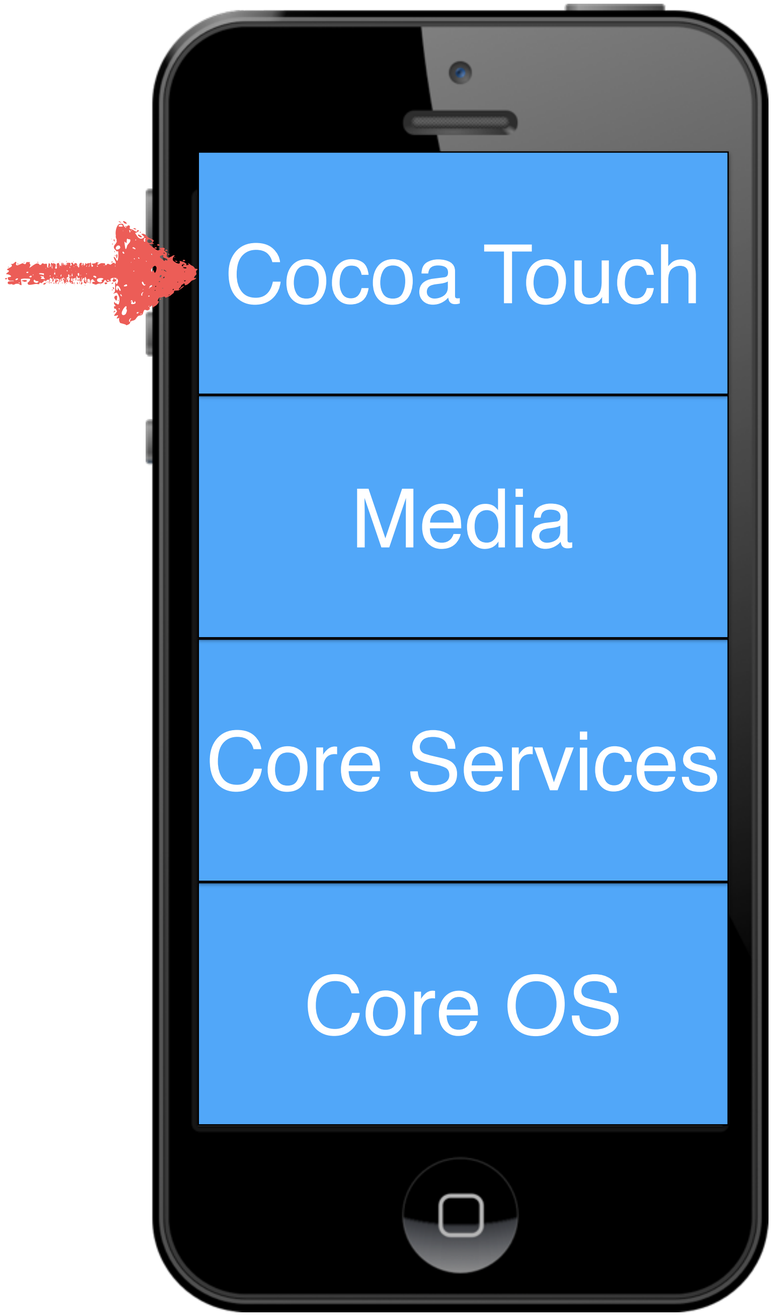
Source: Inspired by Stanford - CS193p



What is iOS
3. Media Layer

Core Audio	Images (Png, Jpg…)
OpenAL	PDF
Audio Mixing	Quartz 2D
Audio Recording	Core Animation
Videos	OpenGL ES

Source: Inspired by Stanford - CS193p



What is iOS
4. Cocoa Touch Layer

Multi-Touch

Core Motion

View Hierarchies

Localization

Controls

Alerts

Web View

MapKit

ImagePicker

Camera

Source: Inspired by Stanford - CS193p

What is a Framework

Additional code written by others

Using Maps

- Don’t write low-level code to locate our position
- Using the Mapkit framework to get current locations

Some iOS Frameworks (1/2)

Framework	Prefix	Description
Foundation	NS	Contains interfaces for managing strings, collections, and other low-level data types.
UIKit	UI	Contains classes and methods for the iOS application user interface layer.
CoreGraphics	CG	Contains the interfaces for Quartz 2D
QuartzCore	CA	Contains the **Core Animation** interfaces.
CoreData	NS	Contains interfaces for managing your application's data model.
AVFoundation	AV	Contains Objective-C interfaces for playing and recording audio and video.
AVKit	AV	Contains Objective-C interfaces for playing and recording audio and video

Source: Apple - iOS Frameworks

Some iOS Frameworks (2/2)

Framework	Prefix	Description
CoreBluetooth	CB	Provides access to low-power Bluetooth hardware.
CloudKit	CK	Contains Objective-C interfaces for fetching and saving iCloud data.
CoreLocation	CL	Contains the interfaces for determining a user's location.
MapKit	MK	Contains classes for embedding a map interface into your application and for reverse-geocoding coordinates.
GameKit	GK	Contains interfaces for managing peer-to-peer connectivity.
SpriteKit	SK	Facilitates the creation of sprite-based animations and rendering.
many more…	…	…

Source: Apple - iOS Frameworks

Cocoa Touch Framework

UIKit

- User interface elements (buttons, labels, tables)
- Application runtime
- Event handling
- Hardware APIs
- …

Foundation

- Utility classes
- Collection classes (Array, Dictionary)
- Object wrappers for system services
- …

Source: Apple - iOS Frameworks

MVC - Model-View-Controller 1/13

One of an a many possible ways to write apps

Pattern provides a way to structure an separate our code in different modules

Design Pattern

Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 2/13

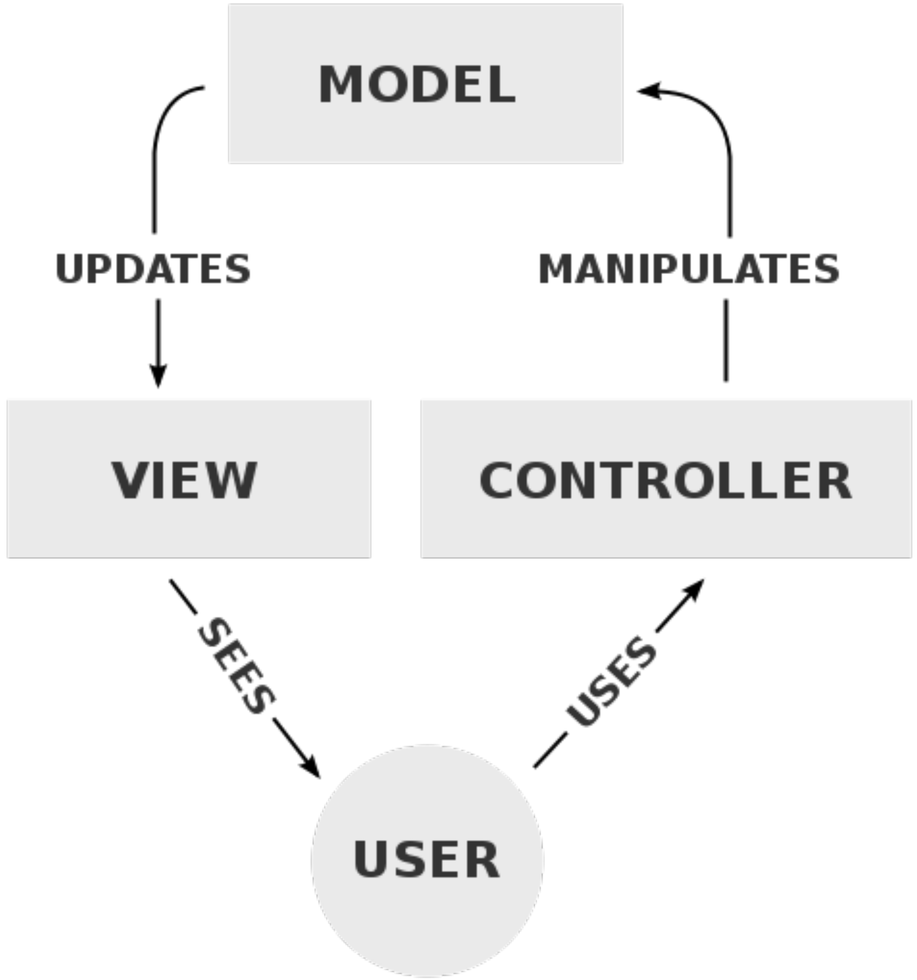
Well-written code is:

- Expensive
- Readable
- Maintainable over Time

Design Pattern

Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 3/13



Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 4/13

Example:
Cat and
Cucumbers

Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 5/13

`Cat`

Attribute	Functions
Name	turn
Direction	accelerate
Type	deccelerate
Location	fireLaser

`Cucumber`

Attribute	Functions
Color	rotate
Size	explode
Radius	implode
Location	bonus

MVC - Model-View-Controller 6/13

What an Users sees

MVC - Model-View-Controller 7/13

Input

Controller
Event Handler

Output

“User clicked on the cat to fired the laster”

“Cat fires the laser”

Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 8/13

What is the position?

Controller
Event Handler

Where is the Cat?

My Position is on 40/100

The Cat is on 40/100

Source: Apple - Start Developing iOS Apps Today

MVC - Model-View-Controller 9/13

Cat

Attribute	Functions
Name	Turn
Direction	Accelerate
Type	Deccelerate
Location	Fire Laser

Model
Often represent as Class

MVC - Model-View-Controller 10/13

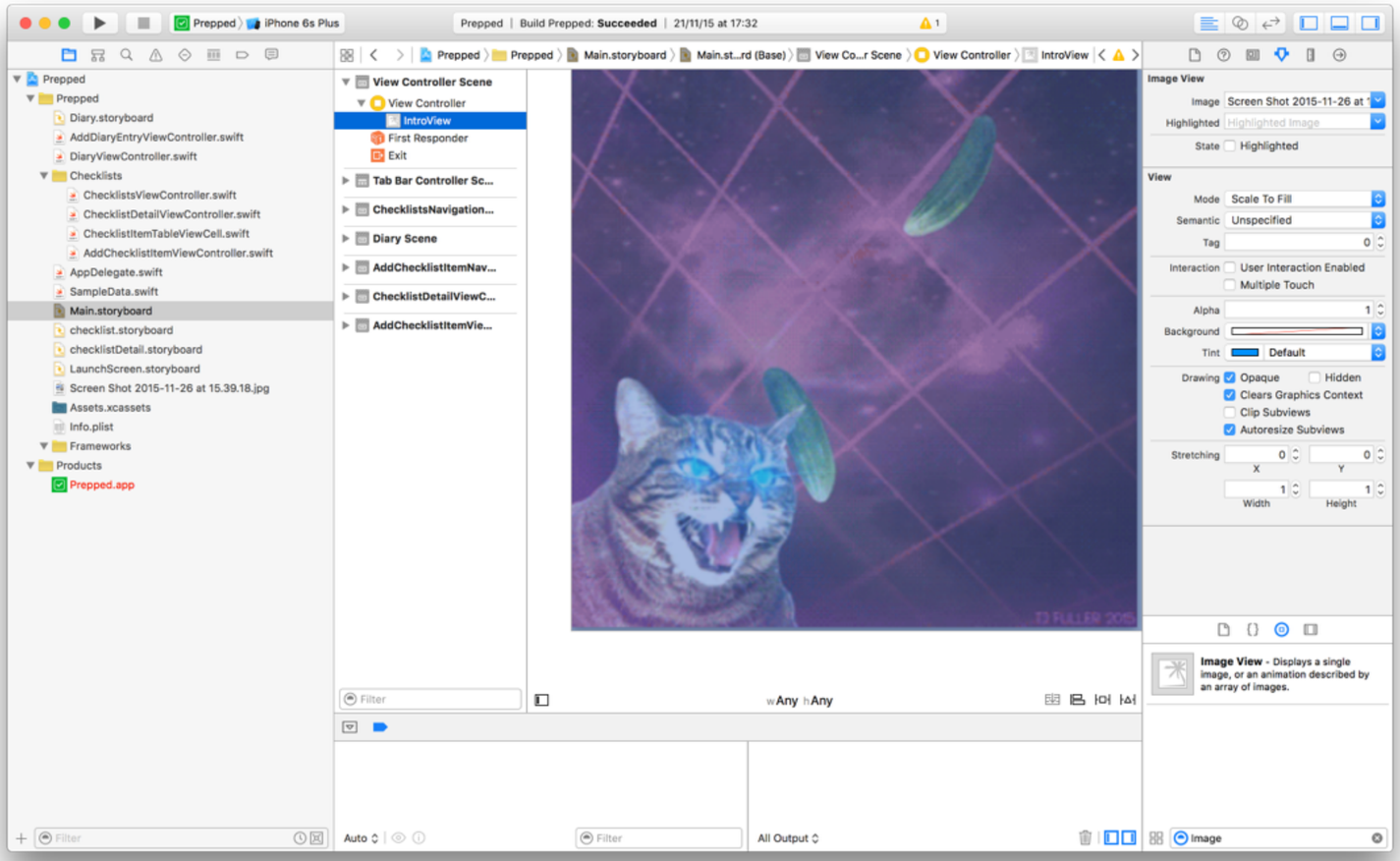
Cat

Attribute	Functions
Name	turn
Direction	accelerate
Type	deccelerate
Location	fireLaser

Cat.swift

```
import fundation

class Cat{
var name:String?
var direction:String?
var type:String?
var Location:location?

func turn(…){…}
func accelerate(…){…}
func deccelerate(…){…}
func fireLaser(…){…}
```

MVC - Model-View-Controller 11/13



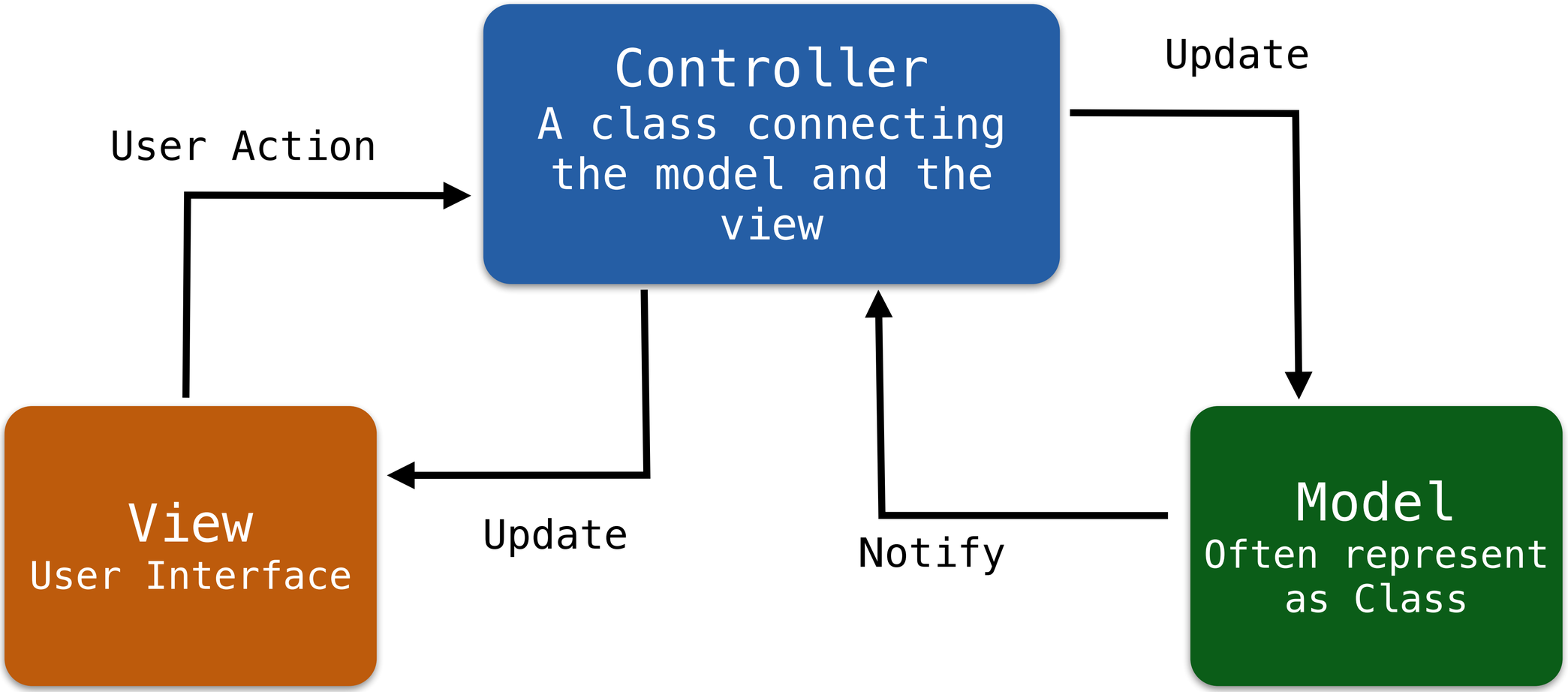
View
User Interface

MVC - Model-View-Controller 12/13

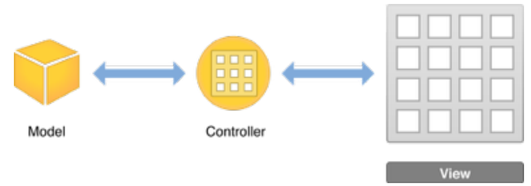
Controller
A class connecting the model and the view

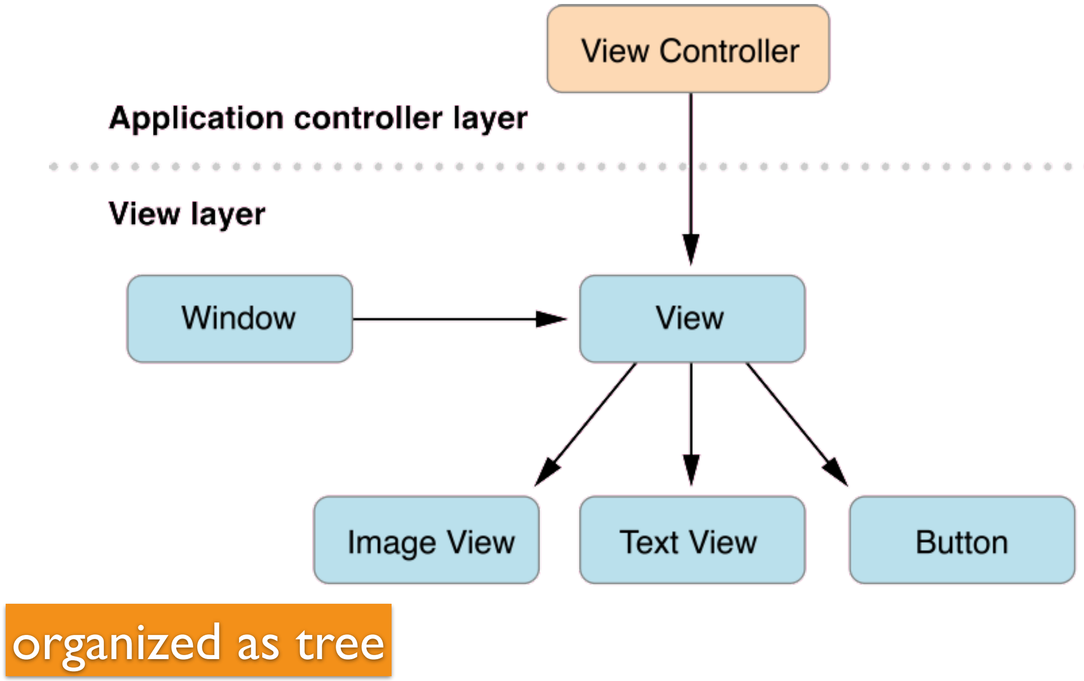
```
import UIKit

class ViewController: UIViewController {

    let cat = Cat()
    var cucumbers = [Cucumbers]()

    override func viewDidLoad() {
        super.viewDidLoad()
       cat.prepareForAction()
       generateCucumbers(cucumbers)
    }

    @IBAction func swipe(sender:
UISwipeG...
```

MVC - Model-View-Controller 13/13

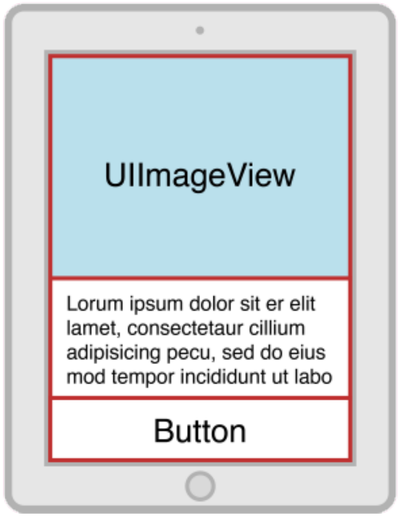


The View Hierarchy



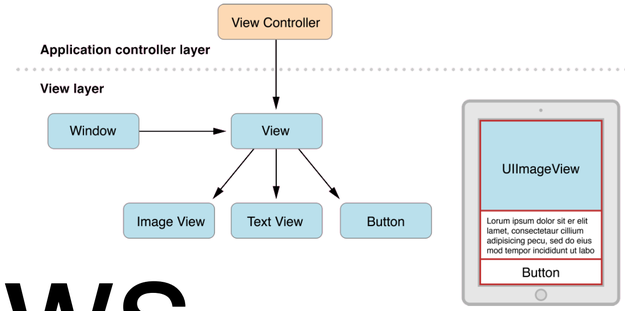


organized as tree



Source: Apple - Start Developing iOS Apps Today

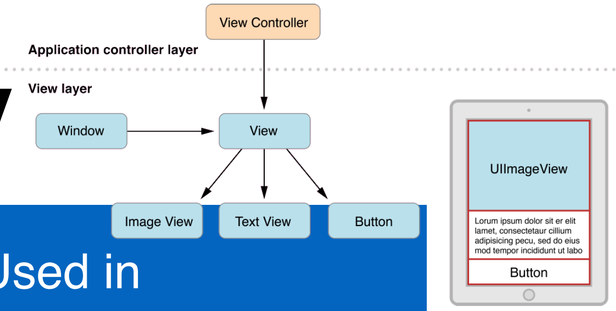
Windows, View Controllers, Views



- **`UIWindow`**
  - Every App has at least one window
  - like a blank window to host view controllers & views
- **`UIViewController`**
  - Manages a self-contained view hierarchy
- **`UIViews`**
  - Draws itself
  - animation
  - Receives events (and forwards them)

Source: Apple - Start Developing iOS Apps Today

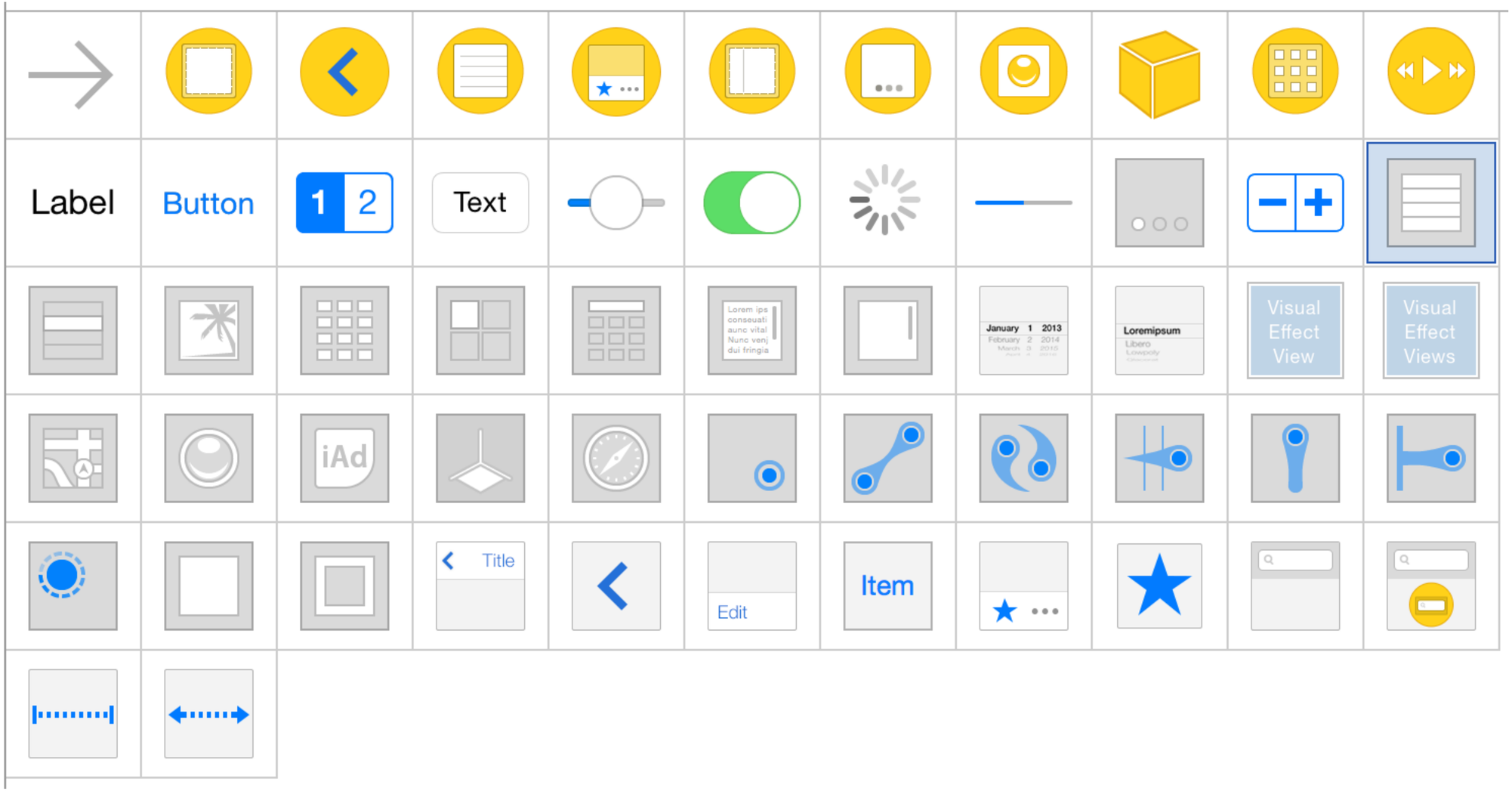
The View Hierarchy



View	Purpose	Used in
Content	Display a particular type of content, such as an image or text.	UIImageView, UILabel
Collections	Display collections or groups of views.	UITableView, UICollectionView
Controls	Perform actions or display information.	UIButton, UISlider, UISwitch
Bars	Navigate, or perform actions.	UIToolbar, UITabbar, UINavigationBar
Input	Receive user input text.	UISearchBar, UITextView
Containers	Serve as containers for other views.	UIView, UIScrollView
Modal	Interrupt the regular flow of the app to allow a user to perform an action.	UIActionSheet, UIAlertView

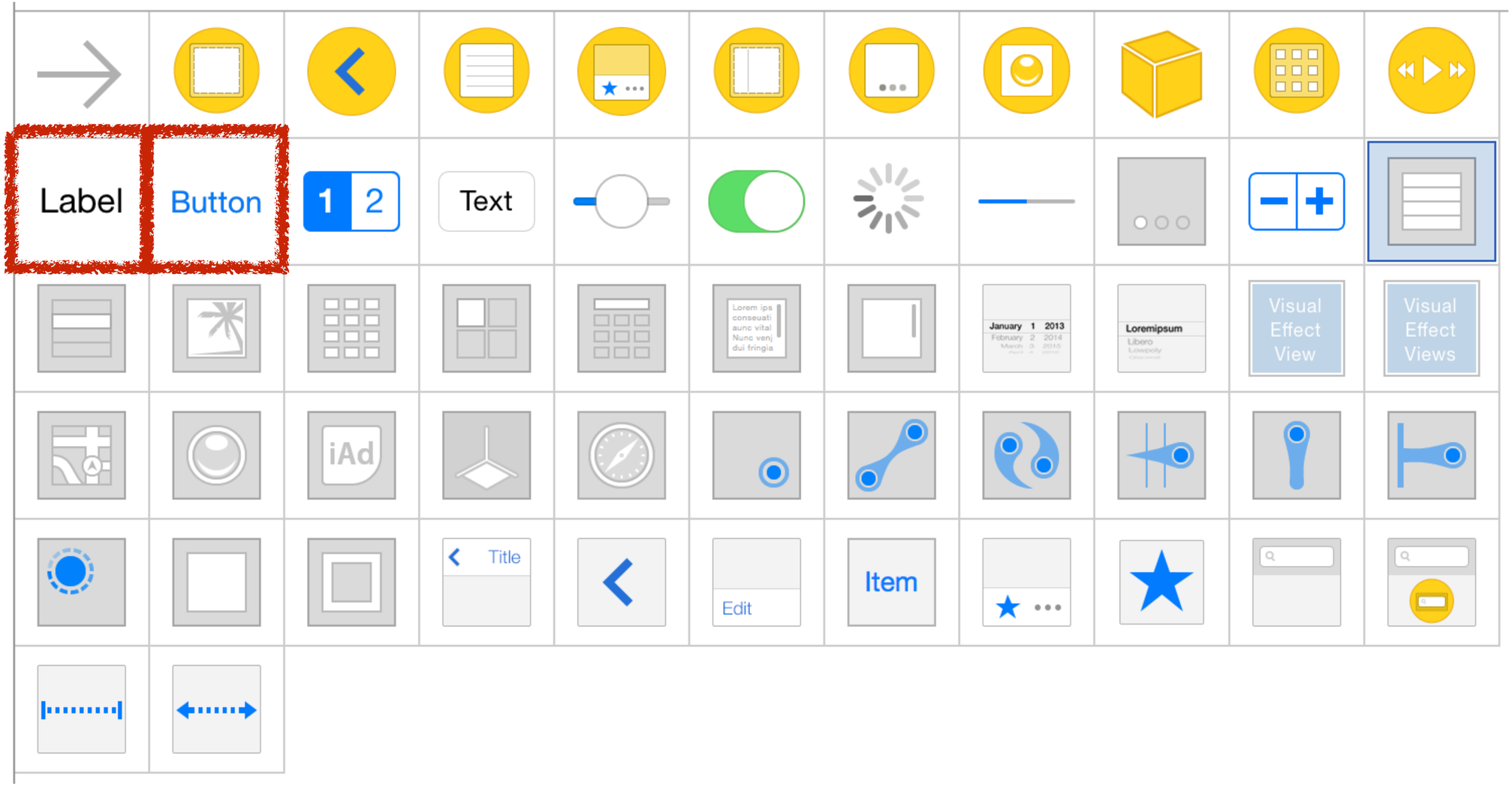
Source: Apple - Start Developing iOS Apps Today

The View Hierarchy



Source: Apple - Start Developing iOS Apps Today

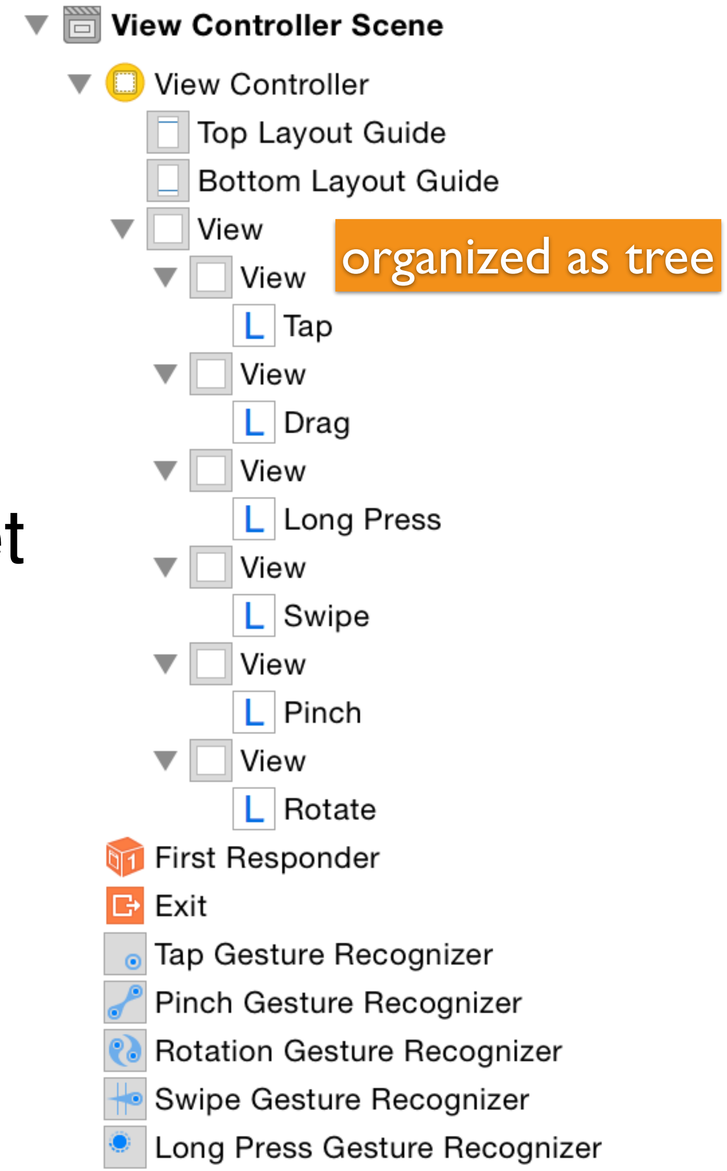
The View Hierarchy



Source: Apple - Start Developing iOS Apps Today

The View Hierarchy

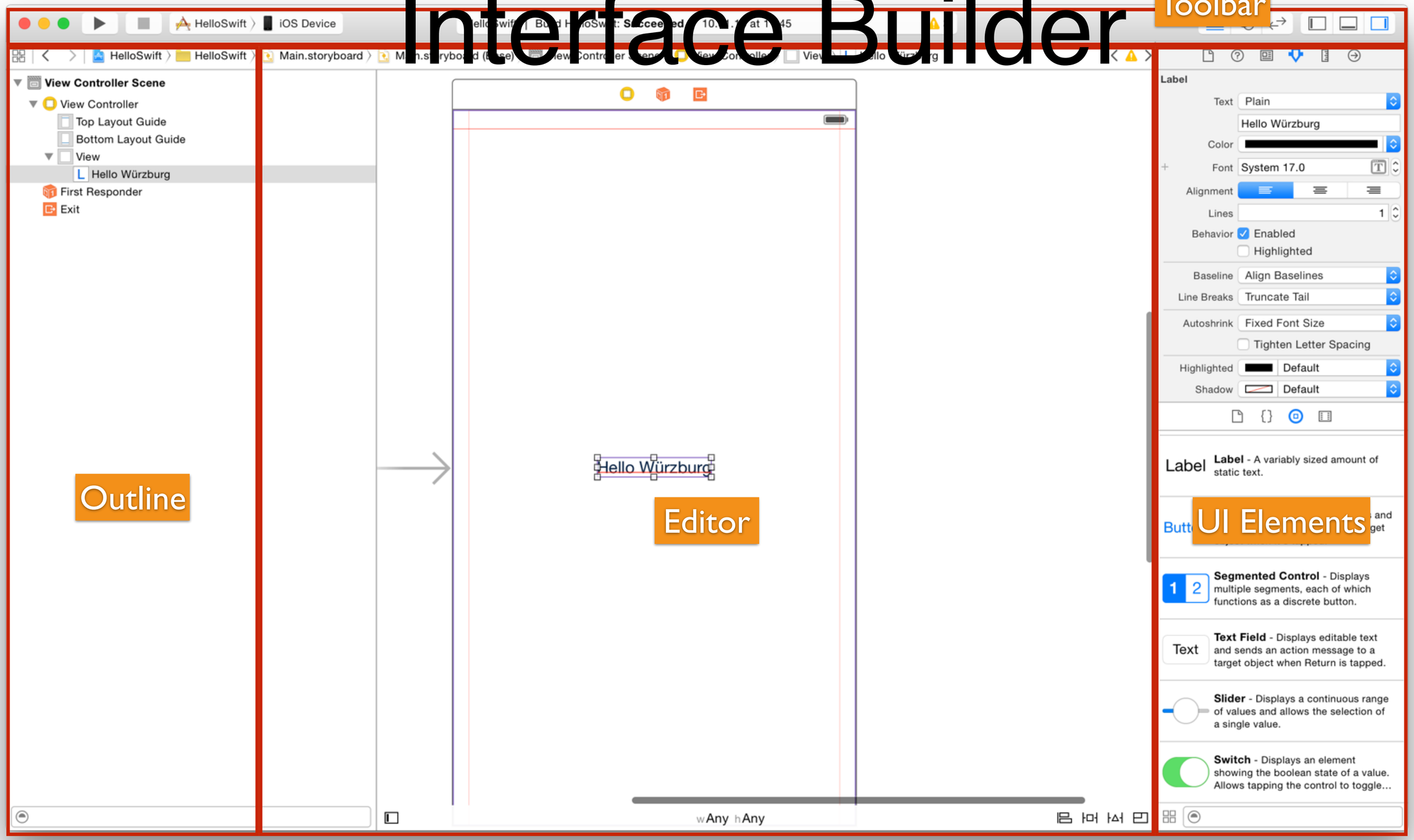
- SDK provides many types of Views to show content
- The Views are organized as tree
- Interface Builder is used to design UI in Storyboards or XIBs
- and connect the UI to code (IBAction, IBOutlet & IBOutletCollection)
  but can be build in Code, too
- View layout is determined by Auto Layout constraints



Interface Builder

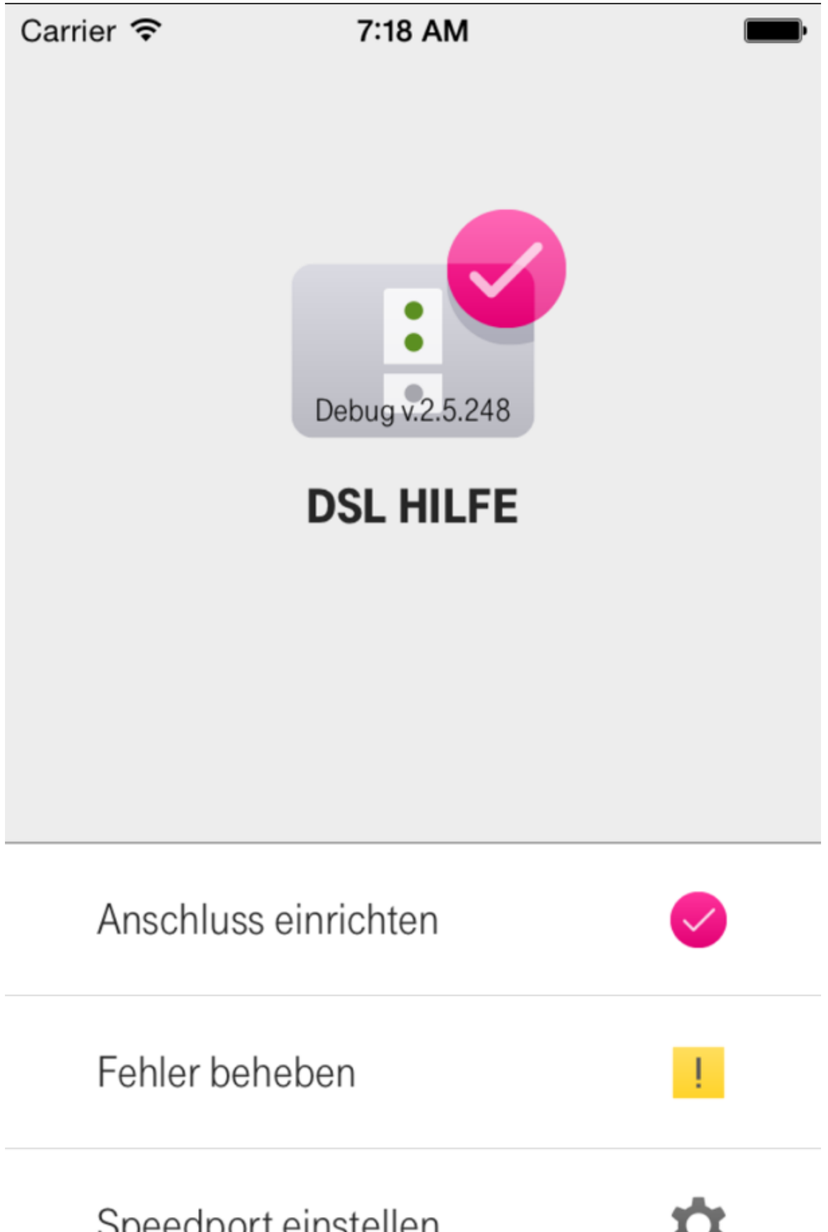
- is a Graphical tool to layout your User Interface
- is optional (you can layout all your Views in Code only if you prefer)
- is used to place UI elements
- sets attributes of elements
- is used to set connections between elements
- is used to wire elements to code
- stores its information in NIB files (XIB files is a valid name, too)
- uses Auto Layout as preferred layout manager

Interface Builder

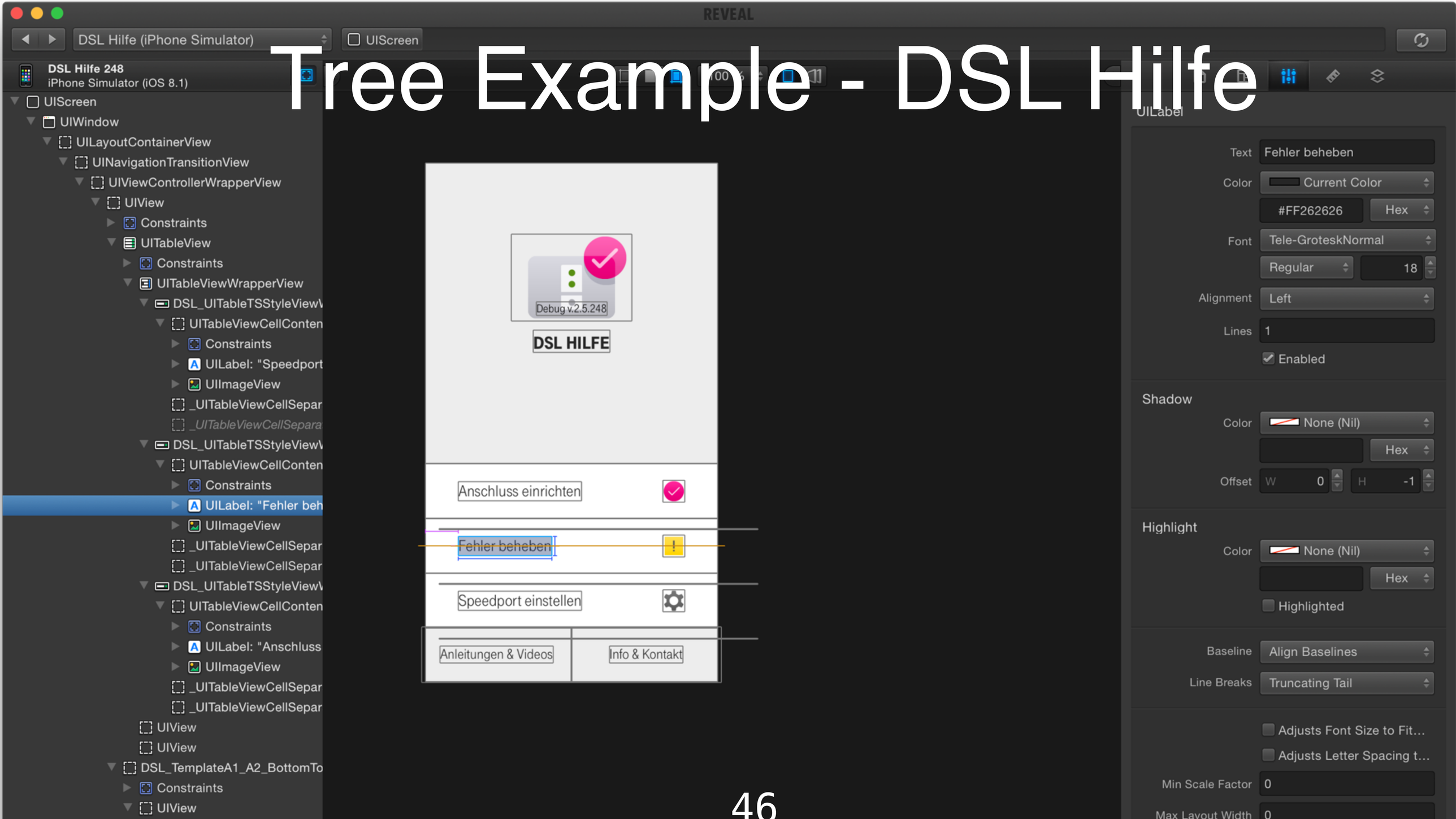


Interface Builder

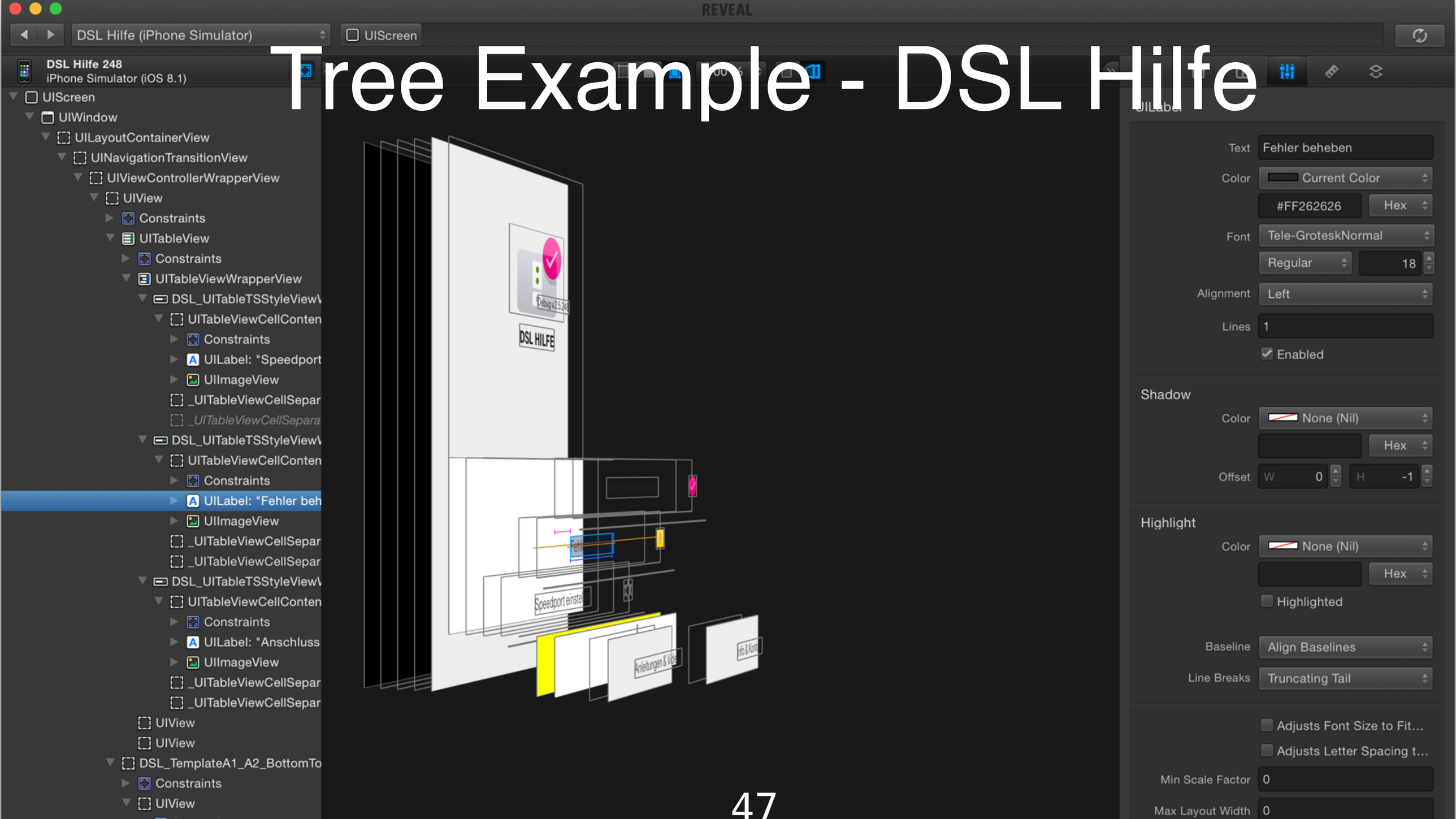
Tree Example - DSL Hilfe



Tree Example - DSL Hilfe



Tree Example - DSL Hilfe

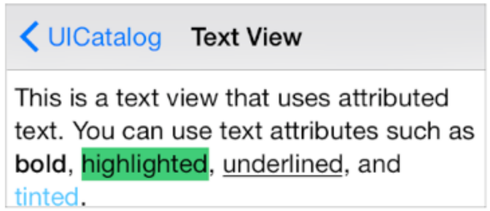


Finding the right View

- Part IV: UI Elements
  - Bars
    - The Status Bar
    - Navigation Bar
    - Toolbar
    - Toolbar and Navigation Bar B…
    - Tab Bar
    - Tab Bar Icons
    - Search Bar
    - Scope Bar
  - Content Views
    - Activity
    - Activity View Controller
    - Collection View
    - Container View Controller
    - Image View
    - Map View
    - Page View Controller
    - Popover
    - Scroll View
    - Split View Controller
    - Table View
    - Text View
    - Web View
  - Controls
    - Activity Indicator
    - Contact Add Button
    - Date Picker
    - Detail Disclosure Button
    - Info Button
    - Label
    - Network Activity Indicator
    - Page Control
    - Picker
    - Progress View
    - Refresh Control
    - Rounded Rectangle Button
    - Segmented Control

Text View

A **text view** accepts and displays multiple lines of attributed text.



API NOTE

To learn more about defining a text view in your code, see Text Views in *UIKit User Interface Catalog*.

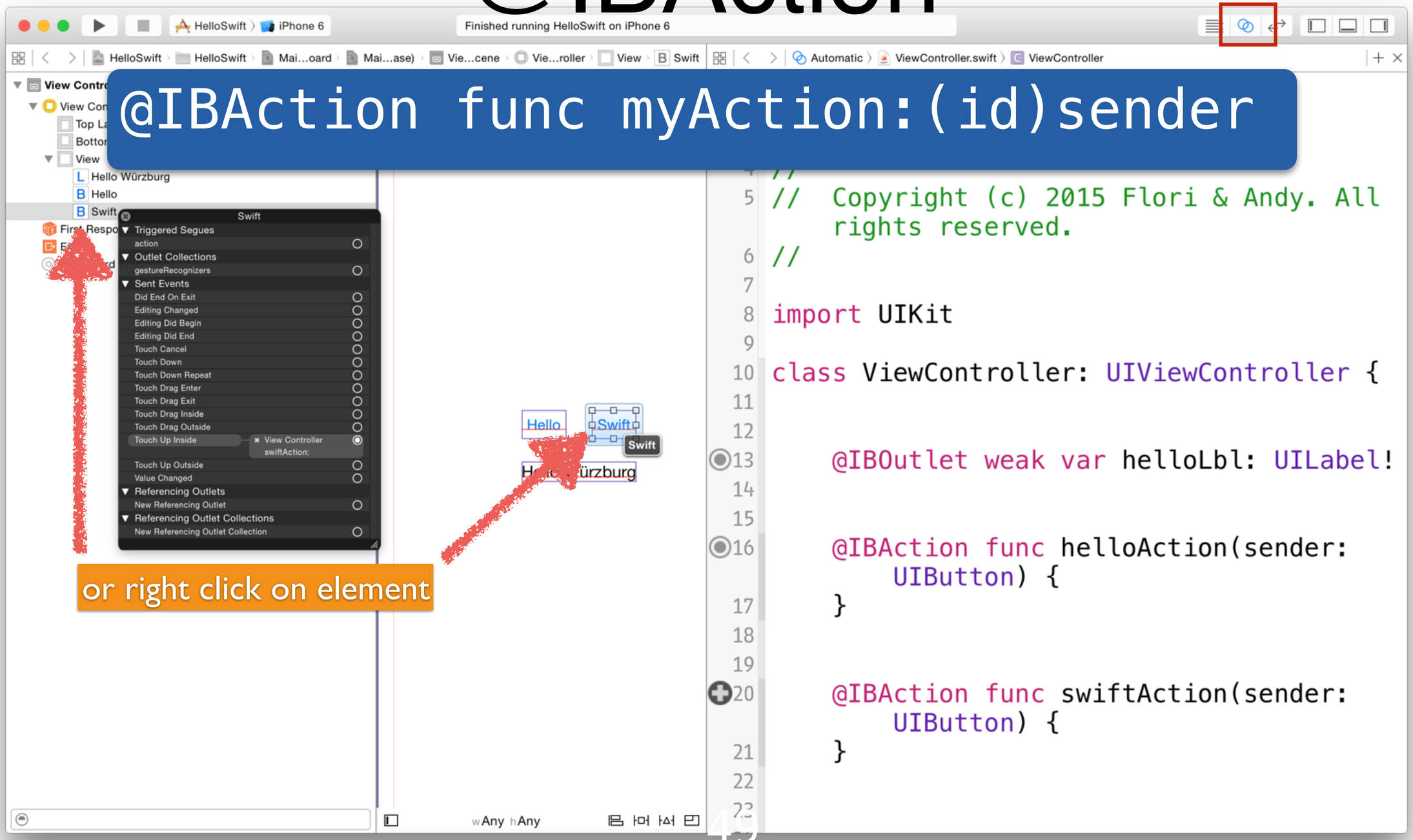
A text view:

- Is a rectangle of any height
- Supports scrolling when the content is too large to fit inside its bounds
- Supports custom fonts, colors, and alignments (by default, a text view displays left-aligned system font in black)
- Can support editing, in which case a keyboard appears when the user taps inside the text view (the keyboard's input method and layout are det
  settings)

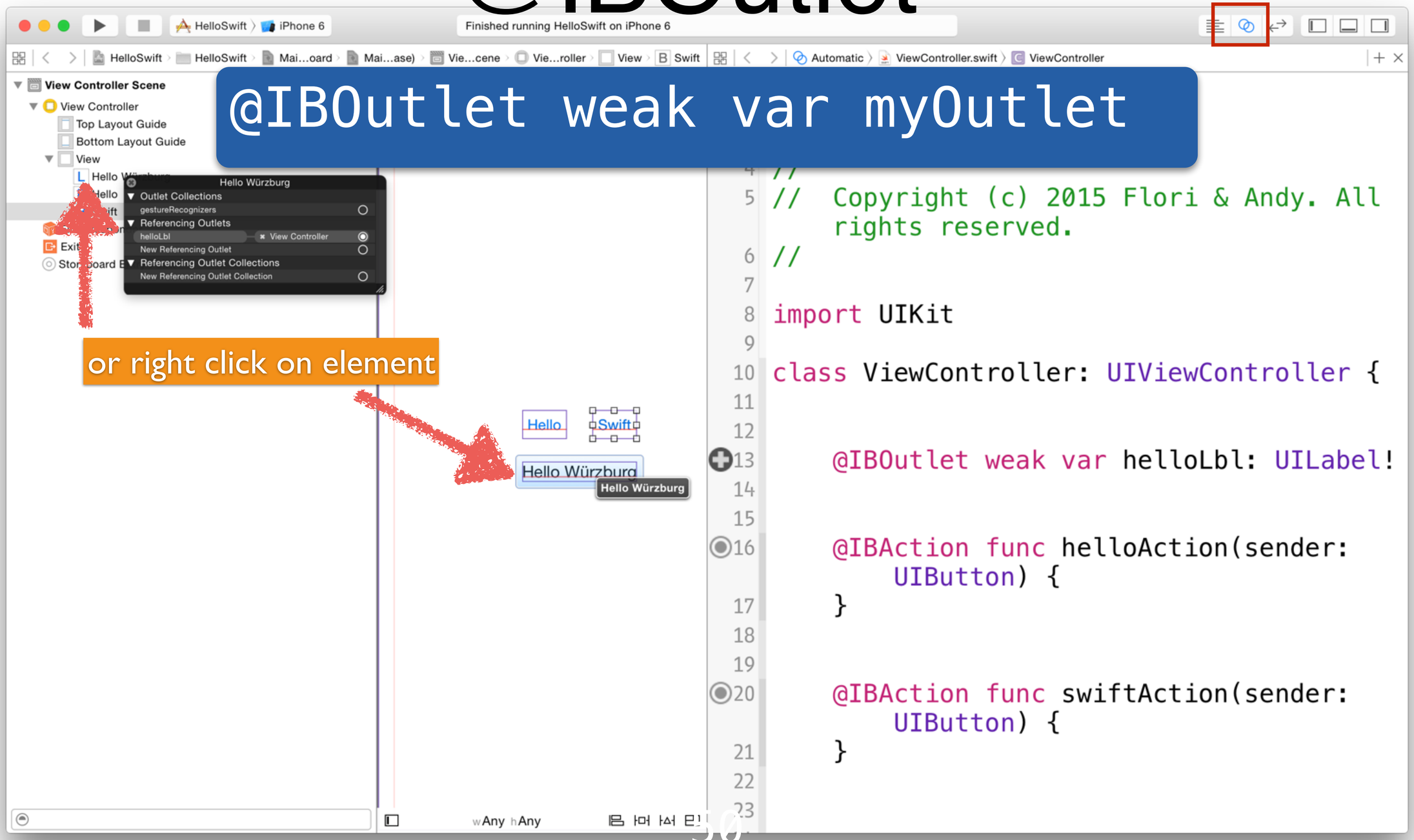
⌘ + Shift + 0 —> "Part IV: UI Elements"

Always make sure the text is easy to read. Although you can use attributed strings to combine multiple fonts, colors, and alignments in creative wa

@IBAction


@IBAction func myAction:(id)sender
Swift
Triggered Segues
action
Outlet Collections
gestureRecognizers
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
View Controller
swiftAction:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
or right click on element
Hello
Swift
Swift
wAny hAny
Copyright (c) 2015 Flori & Andy. All rights reserved.
import UIKit
class ViewController: UIViewController {
@IBOutlet weak var helloLbl: UILabel!
@IBAction func helloAction(sender: UIButton) {
}
@IBAction func swiftAction(sender: UIButton) {
}

@IBOutlet


@IBOutlet weak var myOutlet
or right click on element
Hello
Swift
Hello Würzburg
Hello Würzburg
13 @IBOutlet weak var helloLbl: UILabel!
16 @IBAction func helloAction(sender: UIButton) {
17 }
20 @IBAction func swiftAction(sender: UIButton) {
21 }

Application Components

AppDelegate.swift

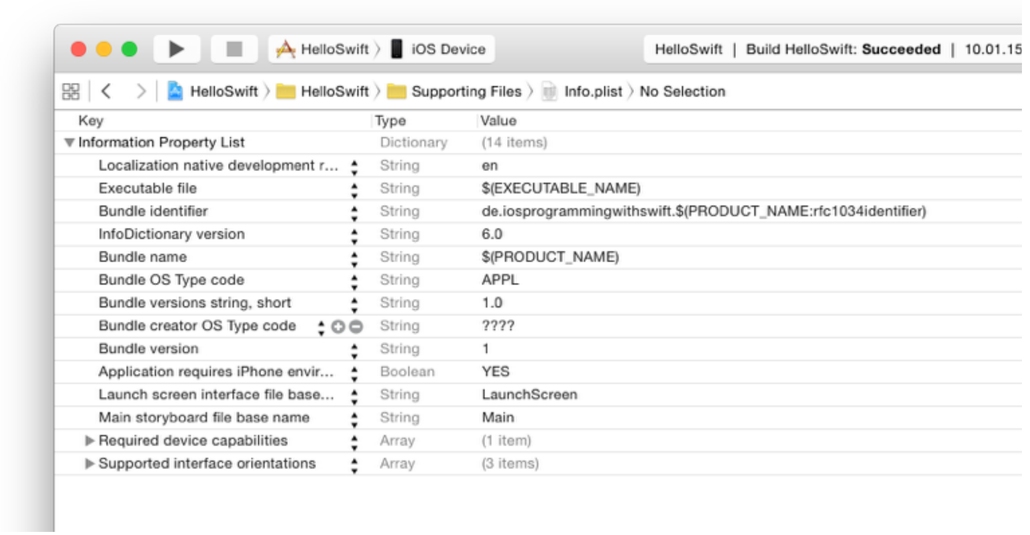
- UIApplication
- CFRunLoop

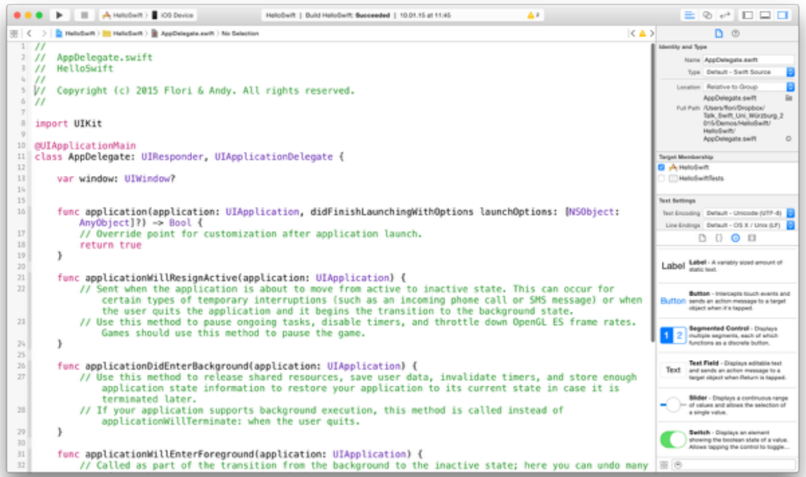
Info.plist

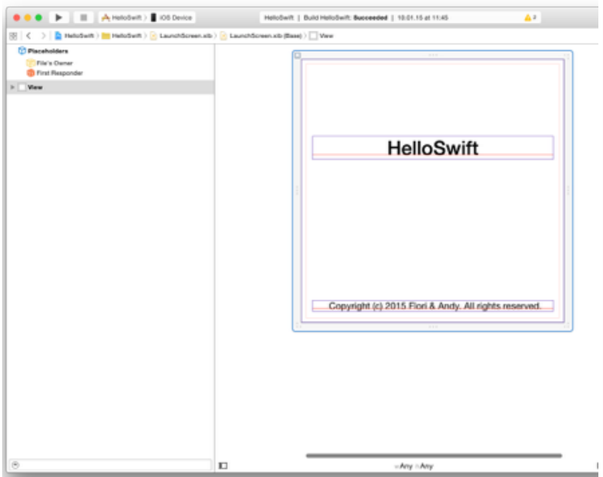
MainStoryboard

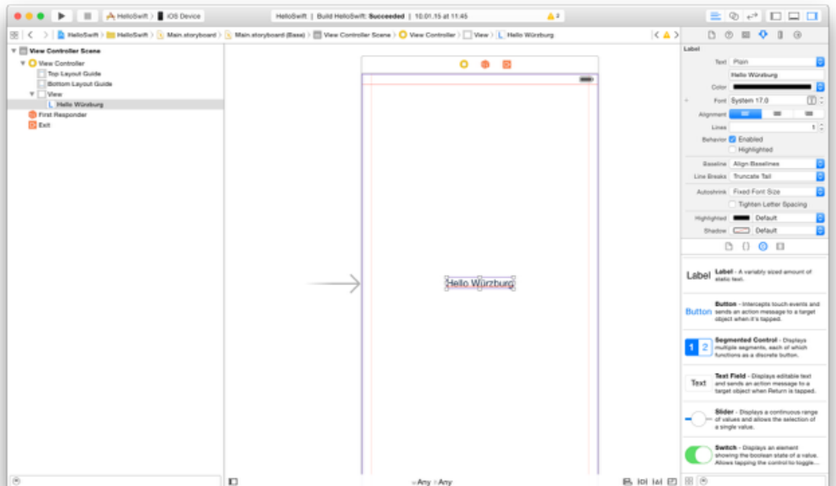
LaunchScreen.xib

ViewController.swift

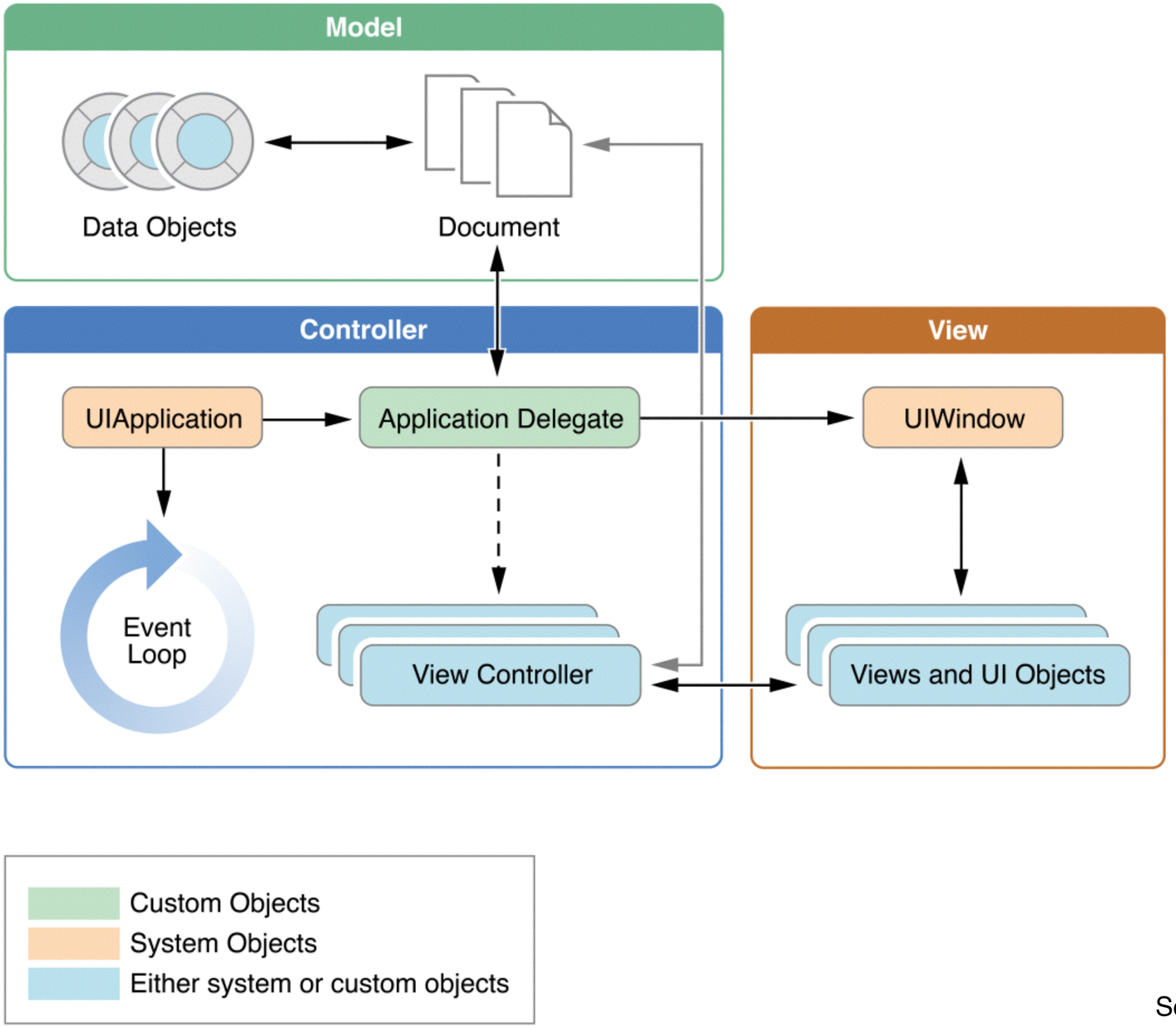






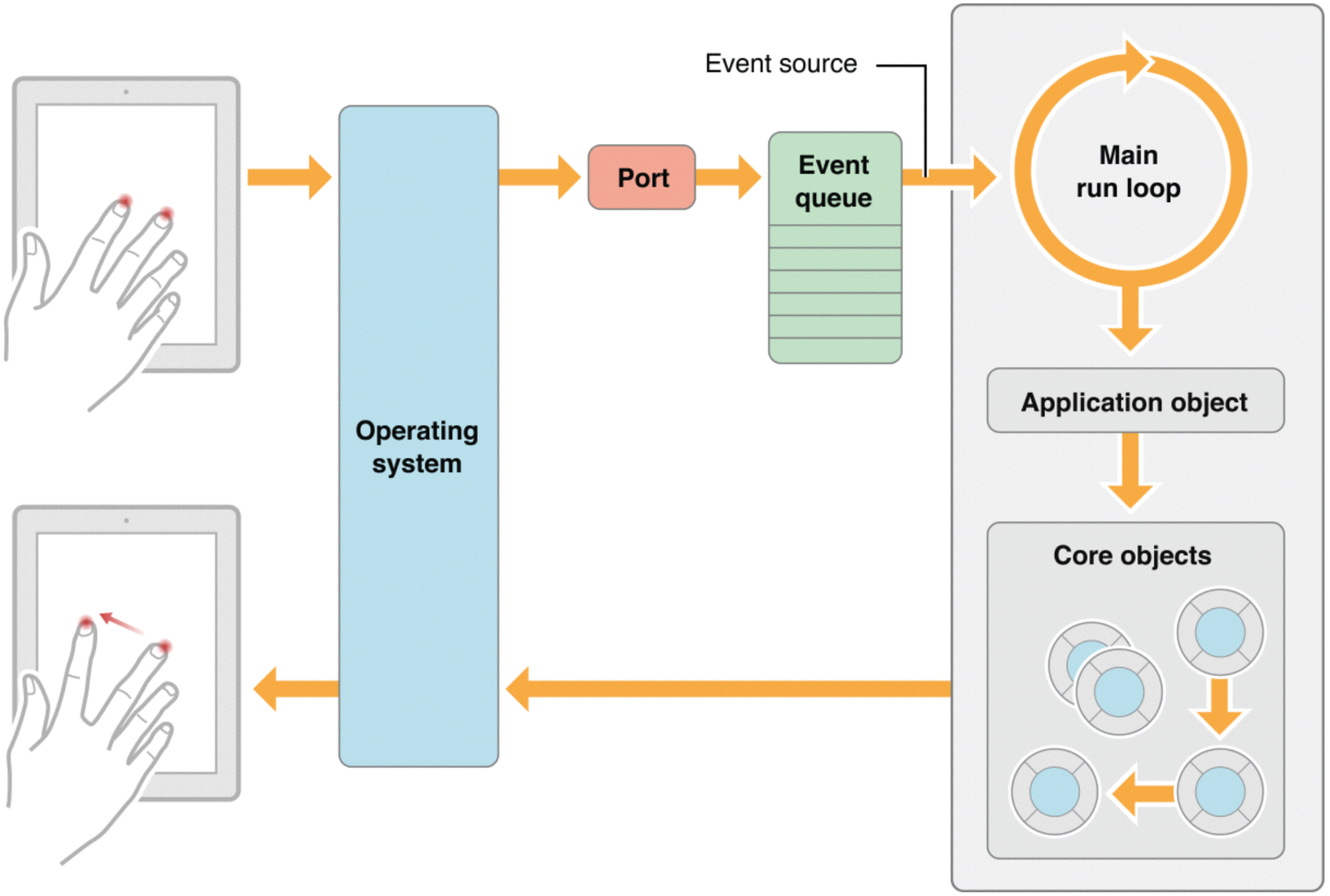


App Life Cycle



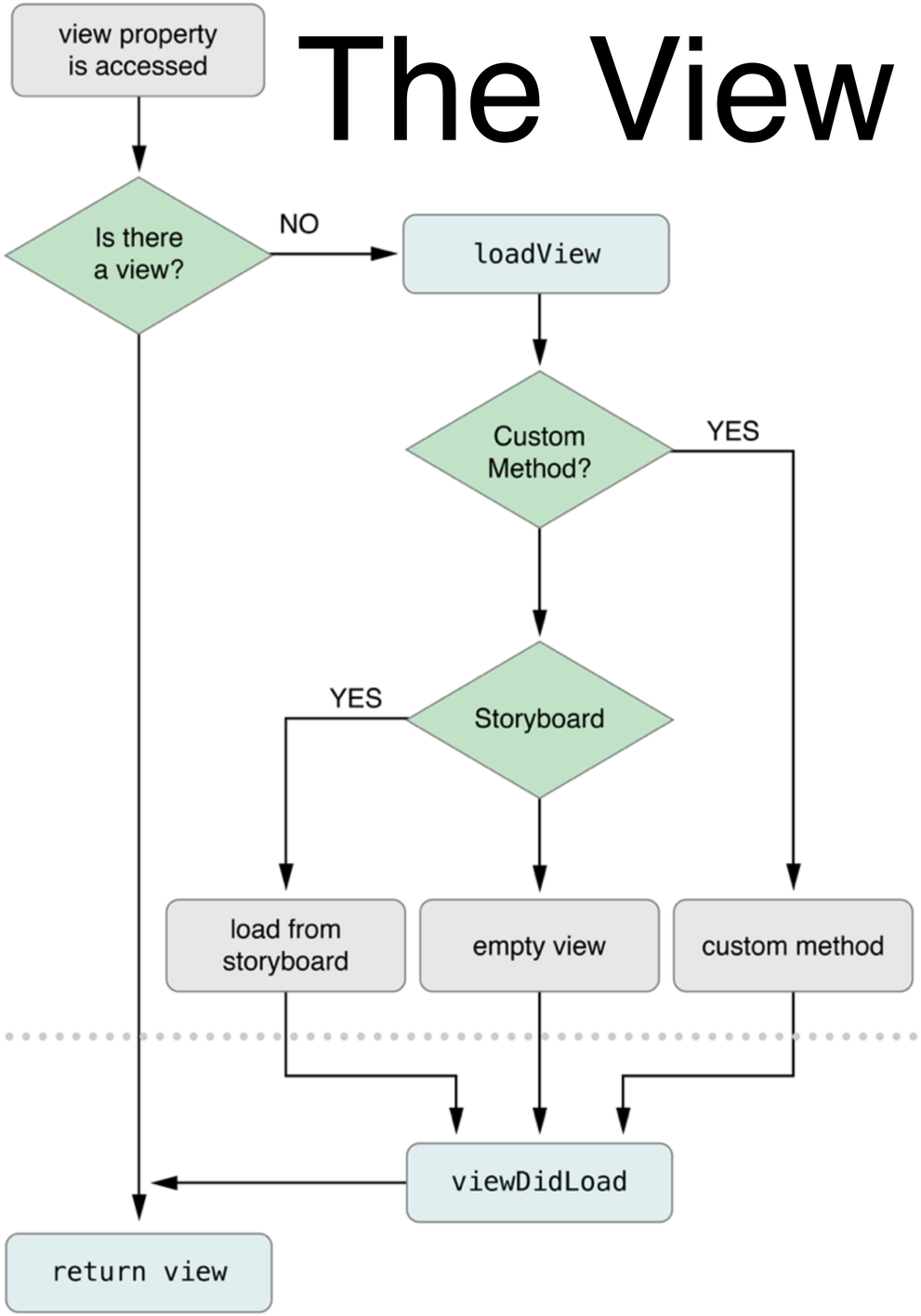
Source: Apple - The App Life Cycle

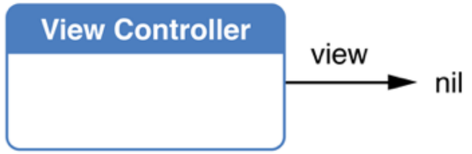
The Main Run Loop

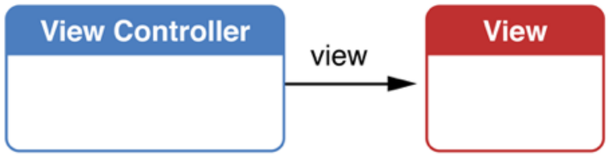


Source: Apple - The App Life Cycle

The View Life Cycle

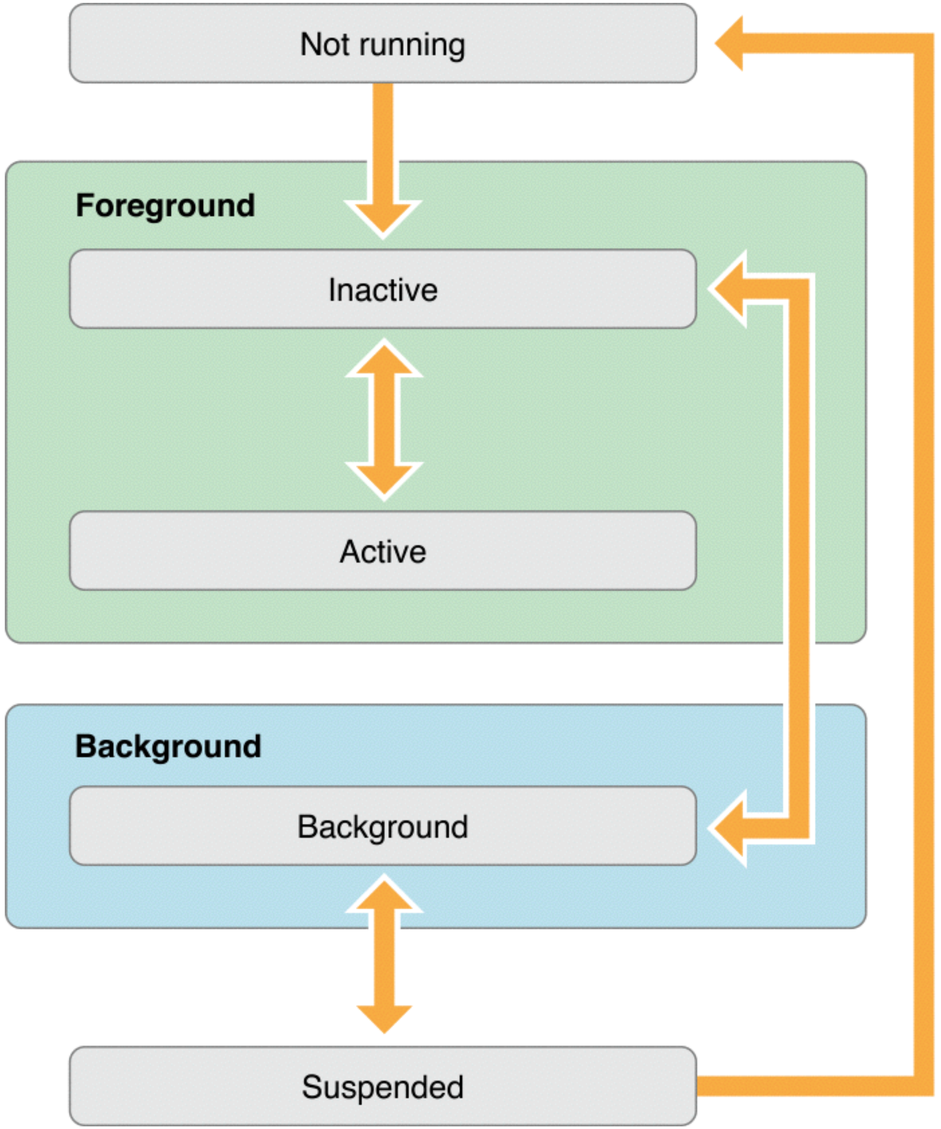






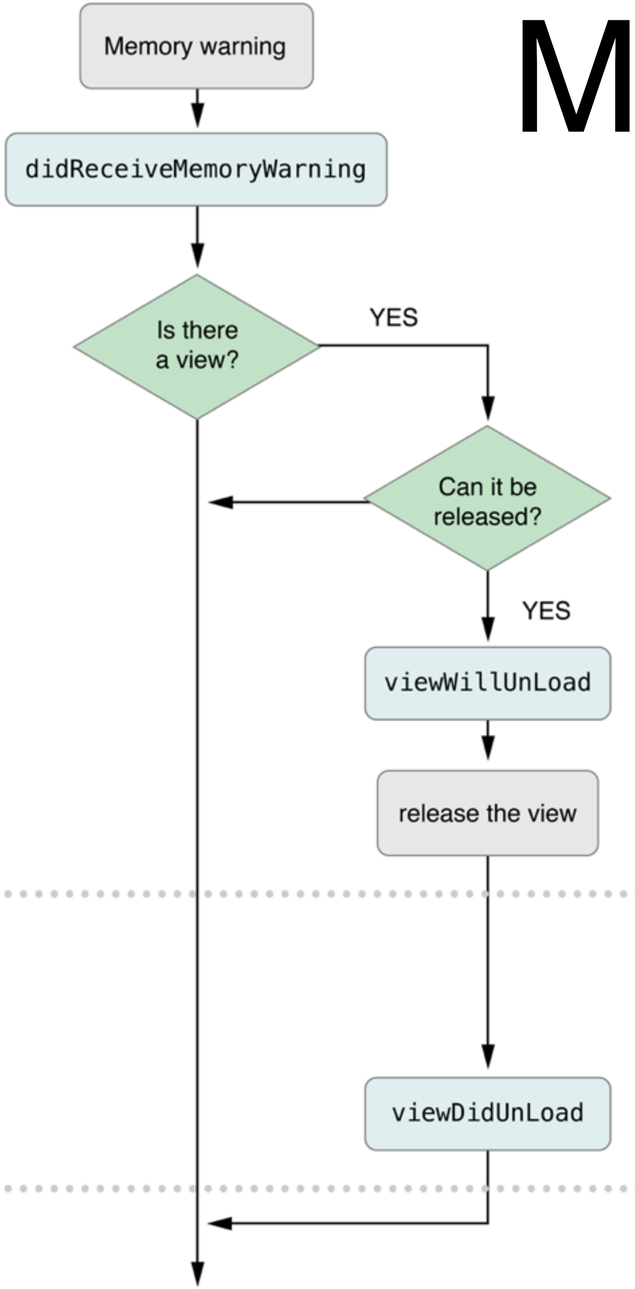
Source: View Controller Programming Guide for iOS

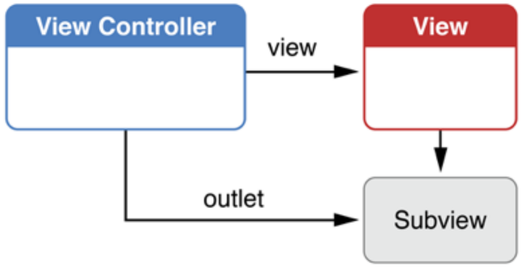
Execution States for Apps



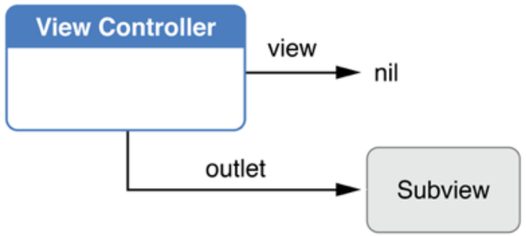
Source: Apple - The App Life Cycle

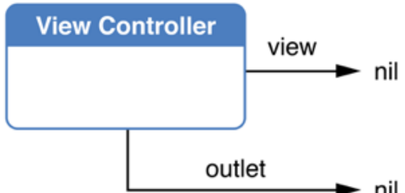
Memory Warnings





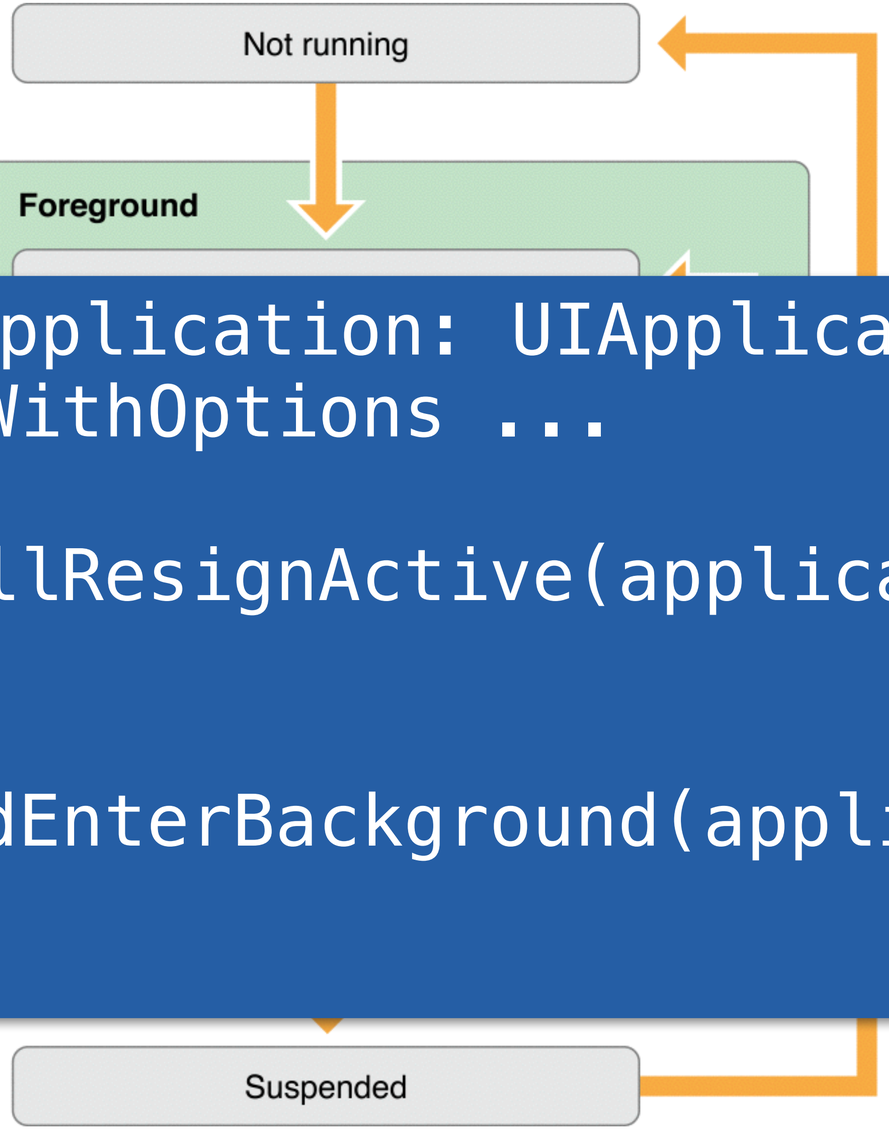

```
override func
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
```





Source: View Controller Programming Guide for iOS

Execution States for Apps

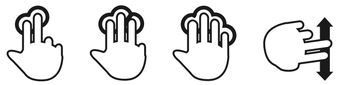


AppDelegate.swift

```
 func application(application: UIApplication,
didFinishLaunchingWithOptions ...

func applicationWillResignActive(application:
UIApplication) {}

func applicationDidEnterBackground(application:
UIApplication) {}
```

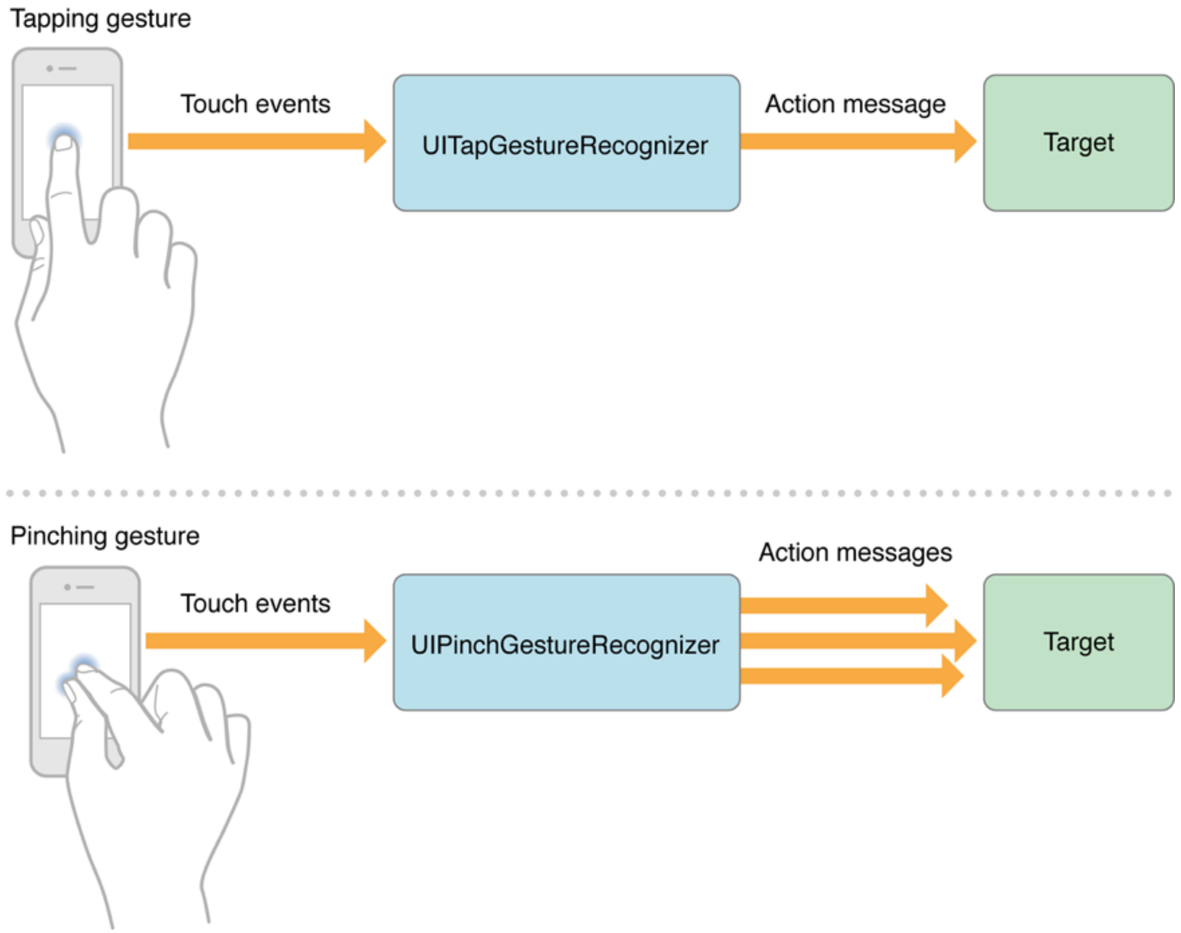
Source: Apple - The App Life Cycle

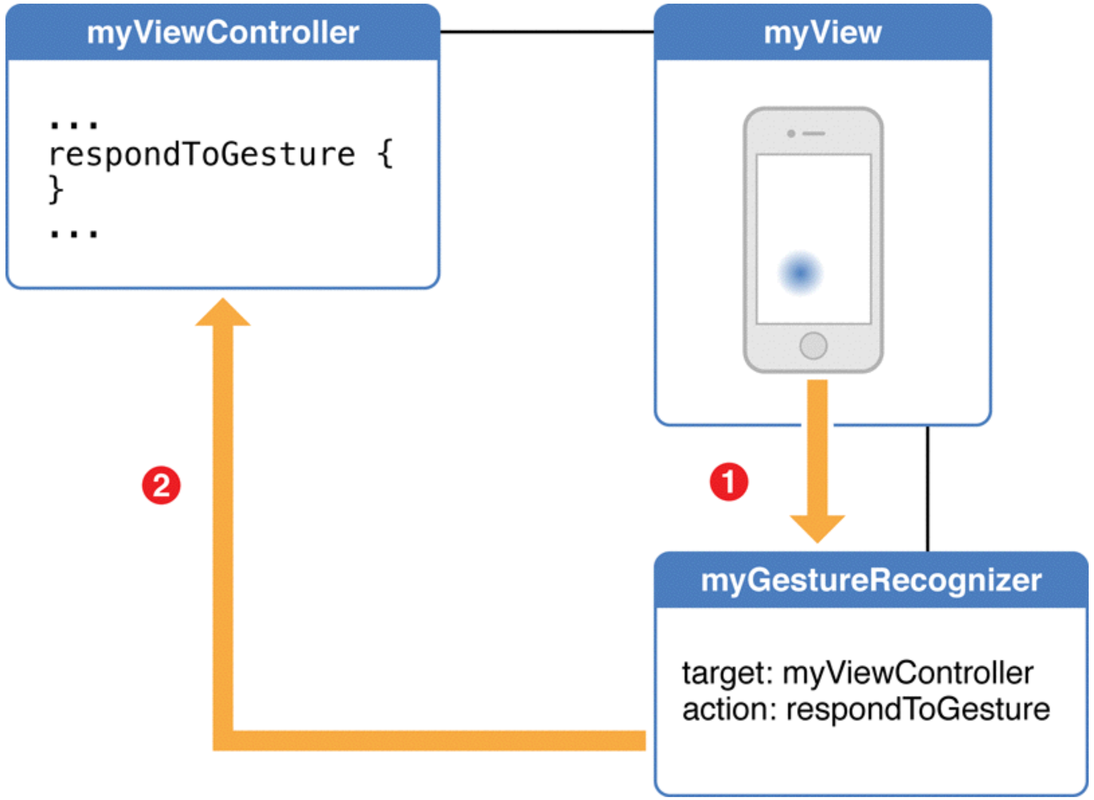
Standard Gestures

View	Purpose
 Tap	To press or select a control (or item)
 Double tap	To zoom in/out and center a block of content or an image
 Touch & hold	In editable text, to change the cursor position
 Swipe/Pan	To scroll or pan (In a table-view row, to reveal the additional buttons)
 Flick	To Scroll or pan quickly
 Pinch open	Zoom in
 Pinch close	Zoom out
	many more

Source: Apple - Start Developing iOS Apps Today

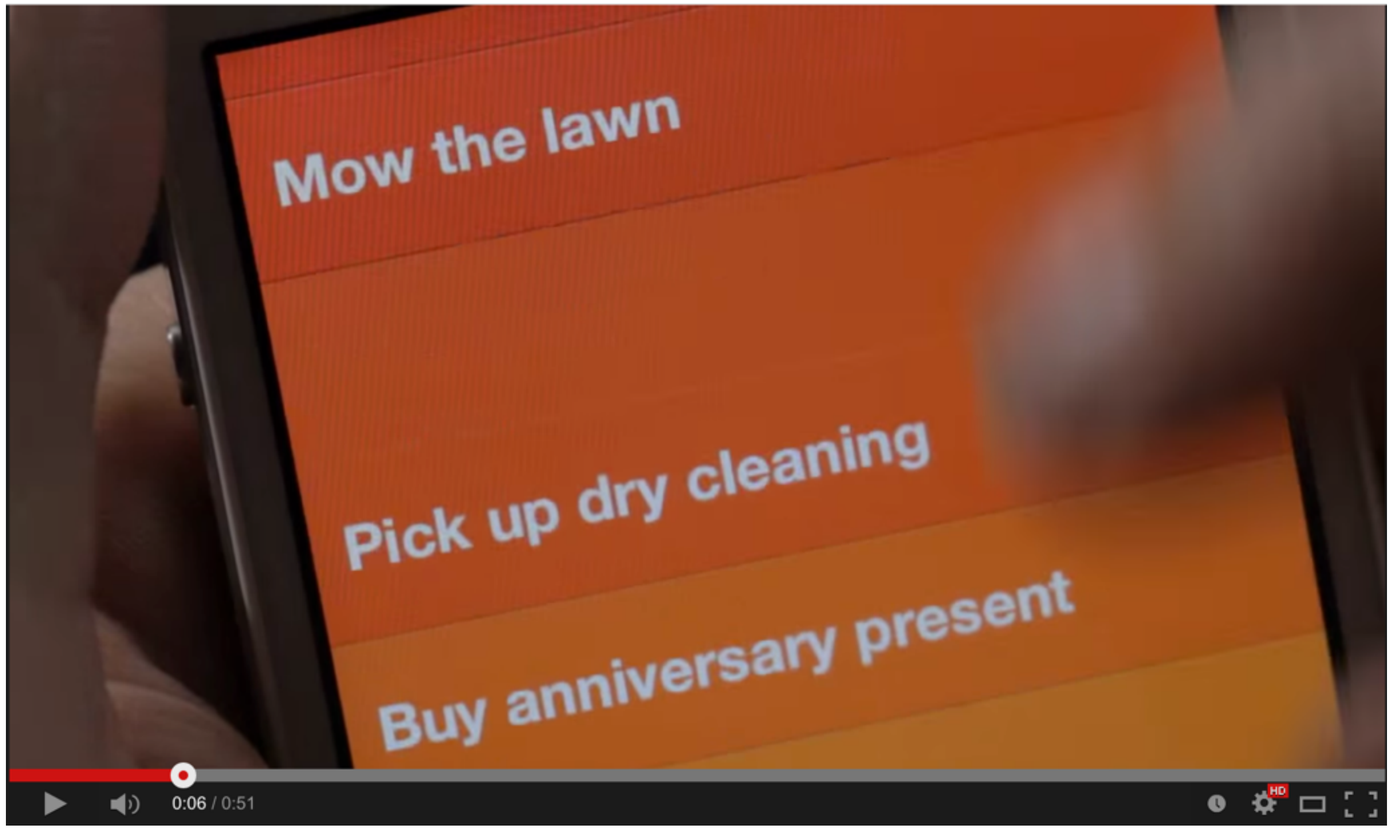
Gesture Recognizers





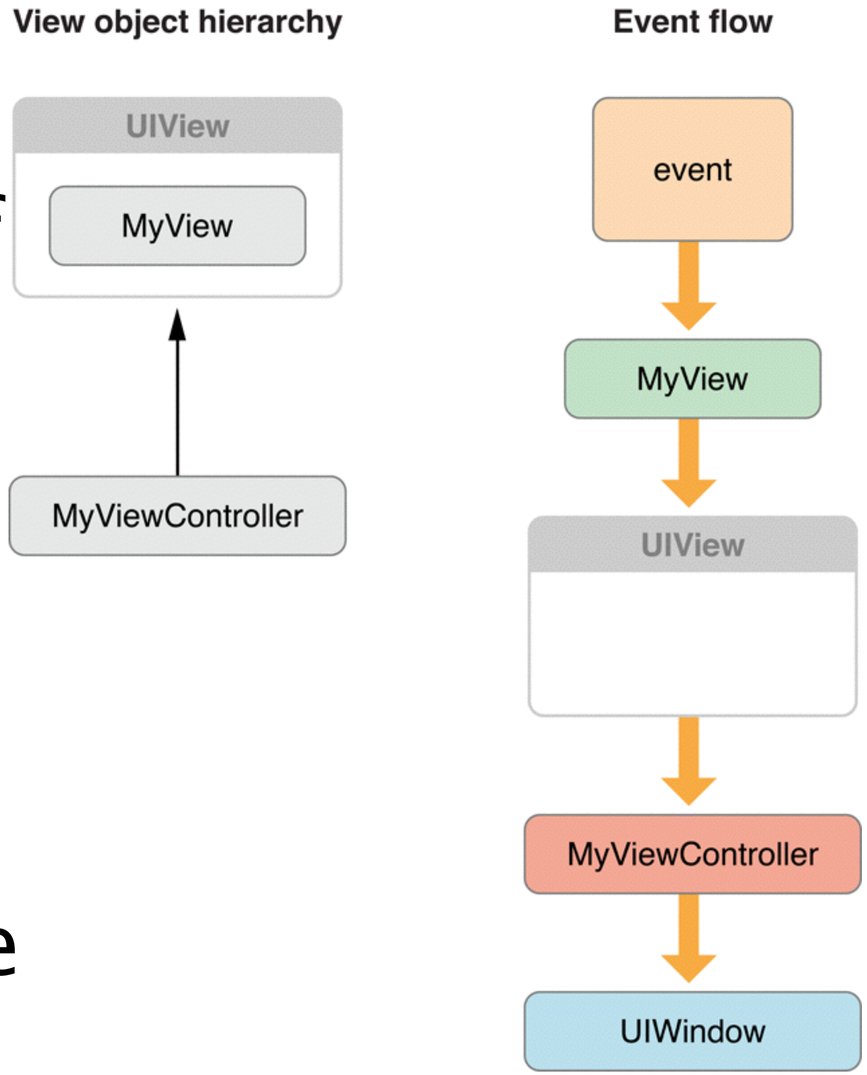
Source: Event Handling Guide for iOS

Clear App


Mow the lawn
Pick up dry cleaning
Buy anniversary present
0:06 / 0:51
HD

Responder Chain (1/2)

- Defines How Events Are Propagated to the App
- The responder chain is a series of linked responder objects.
- It starts with the first responder and ends with the application object.
- If the first responder cannot handle an event, it forwards the event to the next responder in the responder chain.

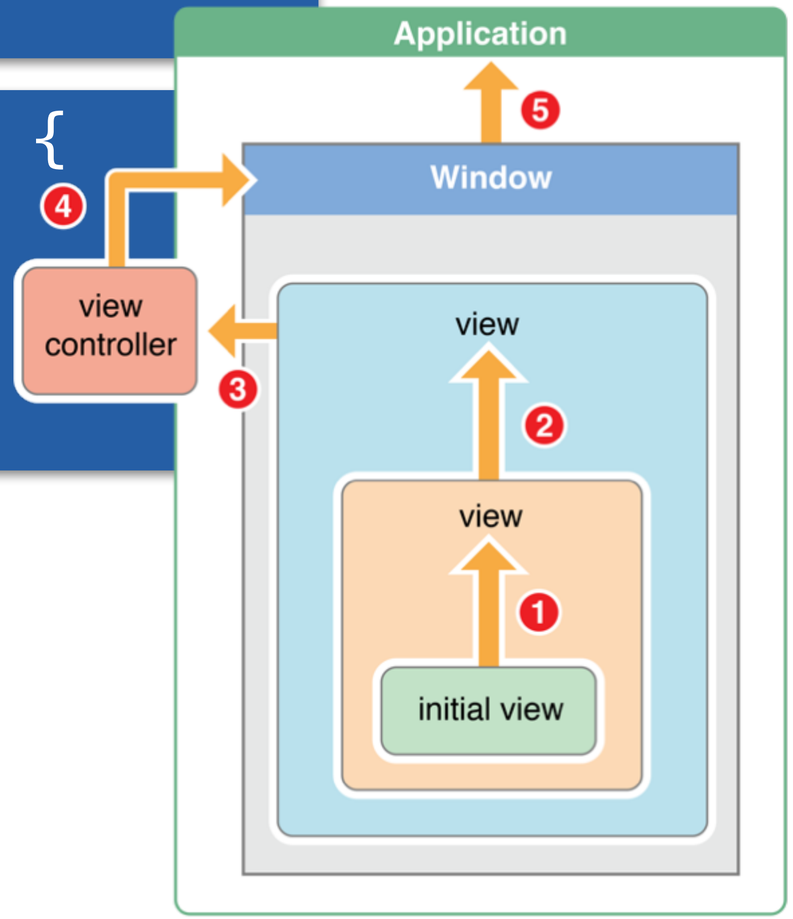


Source: Using View Controllers in the Responder Chain

Responder Chain (2/2)

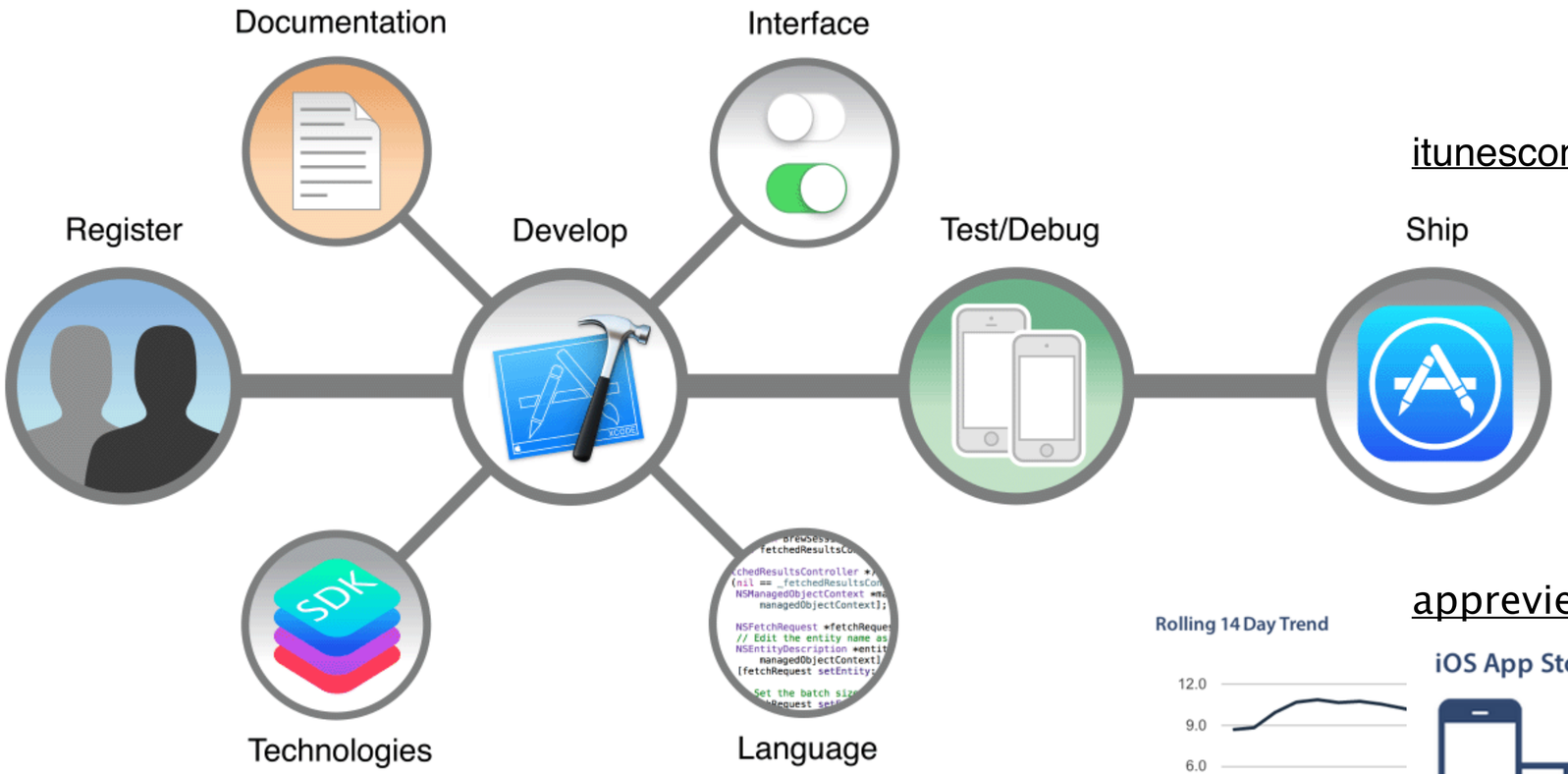
```
override func canBecomeFirstResponder() -> Bool {
    return true
}
```

```
override func viewDidAppear(animated: Bool) {
    super.viewDidAppear(animated)
    self.becomeFirstResponder()
}
```



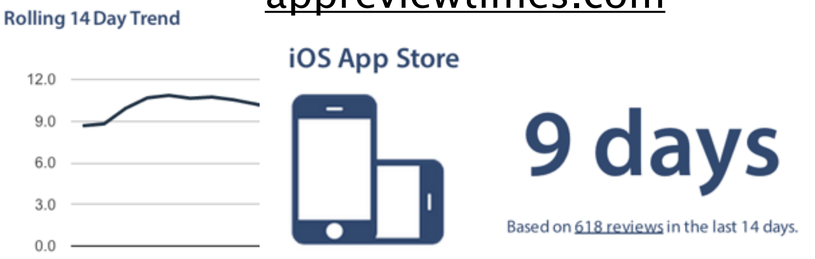
Source: Using View Controllers in the Responder Chain

Overview



itunesconnect.apple.com

appreviewtimes.com



Source: Apple - Start Developing iOS Apps Today